UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Primož Debenec

Inverzna kinematika robotske roke OWI-535

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Danijel Skočaj

Ljubljana 2015

Besedilo je olikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

OWI-535 je enostavna nizko-cenovna robotska roka, ki jo lahko nadgrajeno krmilimo z osebnim računalnikom. V diplomski nalogi razvijte algoritem za izračun inverzne kinematike za to robotsko roko. Algoritem tudi implementirajte v okolju ROS. Razviti sistem naj tako omogoča nadzorovani premik posameznih segmentov robotske roke na način, da se bo vrh robota nahajal na želeni poziciji v 3-dimenzionalnem prostoru. Delovanje sistema oz. natančnost premika robotske roke tudi eksperimentalno ovrednotite.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Primož Debenec, z vpisno številko **63090080**, sem avtor diplomskega dela z naslovom:

Inverzna kinematika robotske roke OWI-535

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Danijela Skočaja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identične s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 27. marca 2015 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Danijelu Skočaju, ki mi je pomagal pri izdelavi diplomske naloge. Zahvalil bi se tudi diplomantu FRI-ja Anžetu Rezlju, ki mi je s svojimi nadgradnjami robotske roke OWI-535 omogočil, da sem lahko izbral to temo diplomske naloge. Zahvala pa gre tudi družini in drugim, ki so mi ves čas študija stali ob strani in me spodbujali.

Kazalo

Slike

Tabele

Seznam uporabljenih kratic

kratica	angleško	slovensko
FK	forward kinematics	direktna kinematika
IK	inverse kinematics	inverzna kinematika
ROS	robot operating system	robotski operacijski sistem
DOF	degreese of freedom	prostostna stopnja
USB	universal serial bus	univerzalno serijsko vodilo
OS	operating system	operacijski sistem

Povzetek

Glavni namen diplomske naloge je izračun inverzne kinematike za robotsko roko OWI-535. Izračun inverzne kinematike omogoča, da podamo robotski roki lego v tridimenzionalnem prostoru, roka pa bo premaknila vse sklepe na tak način, da se bo vrh robota po premiku nahajal na podani legi. Lega je sestavljena iz pozicije in orientacije, zaradi omejitev roke pa se bomo posvetili predvsem prvi. Narejen je svojevrsten način izračuna inverzne kinematike, ki smo ga najprej teoretično izpeljali, potem pa smo izpeljave prenesli v programsko kodo. Postopek, kako pridemo do rešitve inverzne kinematike, je razdeljen na dva dela. Prvi del je izračun inverzne kinematike, kjer je rezultat množica ustreznih rešitev. Drugi del vsebuje izbor najboljše rešitve iz množice rešitev, ki jo dobimo v prvem delu. Narejena je še nadgradnja, ki zajema tudi orientacijo robotske roke. Robotska roka lahko tako pride do podane točke, obrnjena v predhodno določeni smeri. Programerski del je narejen v programskem jeziku C++, in sicer v ogrodju ROS, ki deluje na operacijskem sistemu Linux. Narejene so bile tudi eksperimentalne meritve, iz katerih je razvidno, kolikšna je dejanska natančnost premikanja robotske roke OWI-535.

Ključne besede: robotika, robotska roka, direktna kinematika, inverzna kinematika, ROS, sklep, segment, Arduino, koordinatni sistem, OWI-535.

Abstract

The thesis aims to calculate the inverse kinematics for the OWI-535 robotic arm. The calculation of the inverse kinematics determines the joint parameters that provide the right pose of the end effector. The pose consists of the position and orientation, however, we will focus only on the second one. Due to arm limitations, we have created our own type of the calculation of the inverse kinematics. At first we have derived it only theoretically, and then we have transferred the derivation into the software code. The process, how to calculate the result of the inverse kinematics, is divided into two parts. The first part consists of the calculation of the inverse kinematics, where the result is the set of possible results. The second part includes the selection of the best results from the set of results we have gotten in the first part. We have also made an upgrade that includes the orientation of the robotic arm. The robotic arm can move to the given point, turned to the preselected direction. The programming part is made in the C++ programming language in the ROS framework that is working on the Linux operating system. We have also made some experimental measurements, where we can see the actual movement accuracy of the OWI-535 robotic arm.

Keywords: robotics, robotic arm, direct kinematics, inverse kinematics, ROS, joint, link, Arduino, coordinate system, OWI-535.

Poglavje 1

Uvod

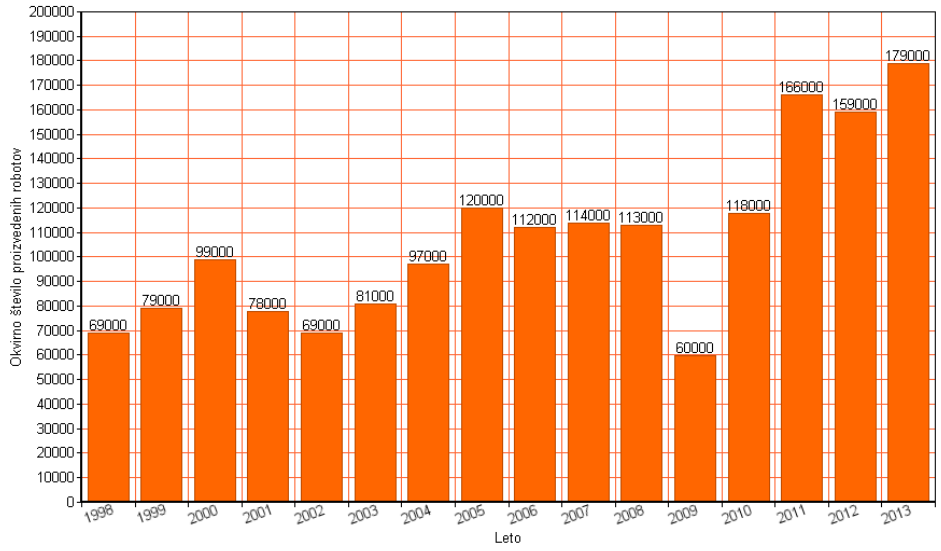
1.1 Motivacija

Roboti so se sprva pojavili v avtomobilski industriji, kjer jih je tudi danes največ. Dandanes se je robotika utrdila v številnih industrijskih panogah. Razlog za vpeljavo robotov v industrijo so predvsem nižji stroški dela ter povečanje kakovosti in zmogljivosti proizvodnje. Dejstvo je, da pri nekaterih opravilih človek ne more biti tako natančen in hiter, kot je lahko robot.

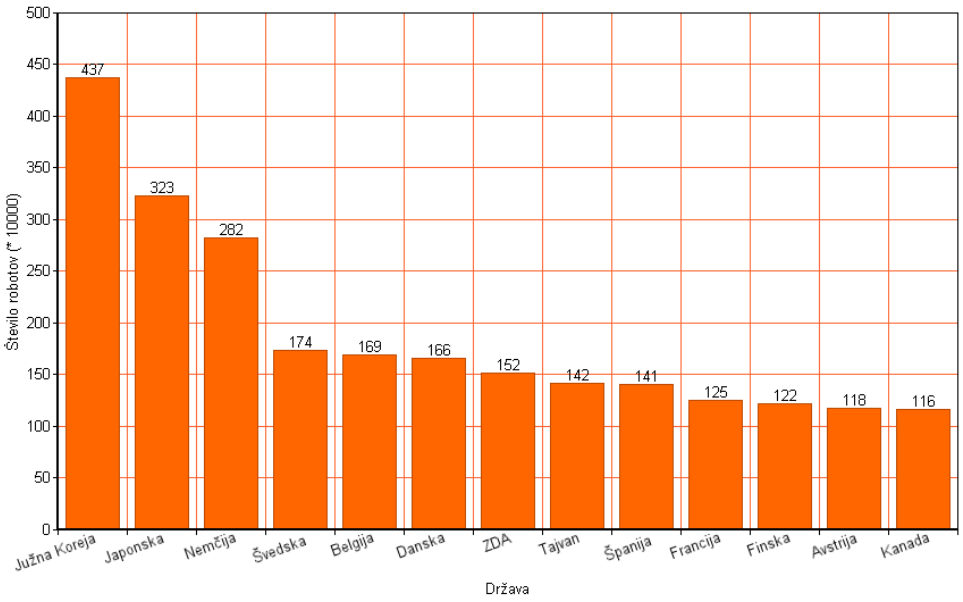
Število letne proizvodnje industrijskih robotov v svetovni industriji [1] se z leti dokaj konstantno povečuje (Slika 1.1). Da ima področje robotike res velik potencial v prihodnosti lahko sklepamo tudi po podatku, da je spletni velikan Google samo v letu 2013 kupil 8 podjetij, ki so specializirana na področju robotike [4].

Po številu robotov glede na število prebivalcev prevladujejo razvitejše Azijske države, sledijo pa jim Evropske države (Slika 1.2). Skupno število delujočih industrijskih robotov na svetu je leta 2013 znašalo okrog 1.5 milijona. Kar tretjino vseh robotov deluje v avtomobilski industriji.

Leta 2013 je imela Slovenija v uporabi nekaj več kot 1600 robotov. To znaša nekaj manj kot 90 robotov na 10.000 zaposlenih, kar je nad svetovnim kot tudi Evropskim povprečjem. Glede na tip robotskih rok v Sloveniji prevladujejo (kot tudi drugje po svetu) artikulirane robotske roke (Slika 1.3), sledijo jim kartezijske in SCARA (angl. Selective Compliance Assembly Robot Arm) robotske roke [1, 2].



Slika 1.1: Okvirna ocena svetovne proizvodnje industrijskih robotov po posameznih letih [2].



Slika 1.2: Statistika števila robotov na 10000 prebivalcev za 13 najvišje uvrščenih držav v letu 2013 [2].

Slika 1.3: Industrijska robotska roka podjetja KUKA Robotics. Slika povzeta po [3].

1.2 Definicija problema in oris rešitve

Inverzna kinematika (IK - angl. inverse kinematics) je določitev vrednosti posameznih sklepov robotske roke, da ustrezajo dani legi vrha roke. Prvi problem pri izračunu IK je, kako izračunati pozicije sklepov robotske roke, da te ustrezajo vrhu roke. Druga, običajno večja težava izračuna IK je, da je vedno mogoče večje število rešitev. Poleg tega je vedno potrebno upoštevati, kakšen je razpon med posameznimi sklepi ter paziti na območja, kamor se roka ne more premakniti, npr. podlaga ali kakšna druga ovira.

Nasprotno od IK deluje direktna kinematika (FK - angl. forward kinematics), ki ima vedno samo eno rešitev. FK je izračun pozicije vrha robotske roke glede na položaje sklepov. FK bomo uporabljali za preverjanje pravilnosti rešitve IK. V tej diplomski nalogi bomo vse izračune in eksperimentalne meritve izvajali na točno določeni robotski roki, in sicer na robotski roki OWI-535 (Slika 1.4).

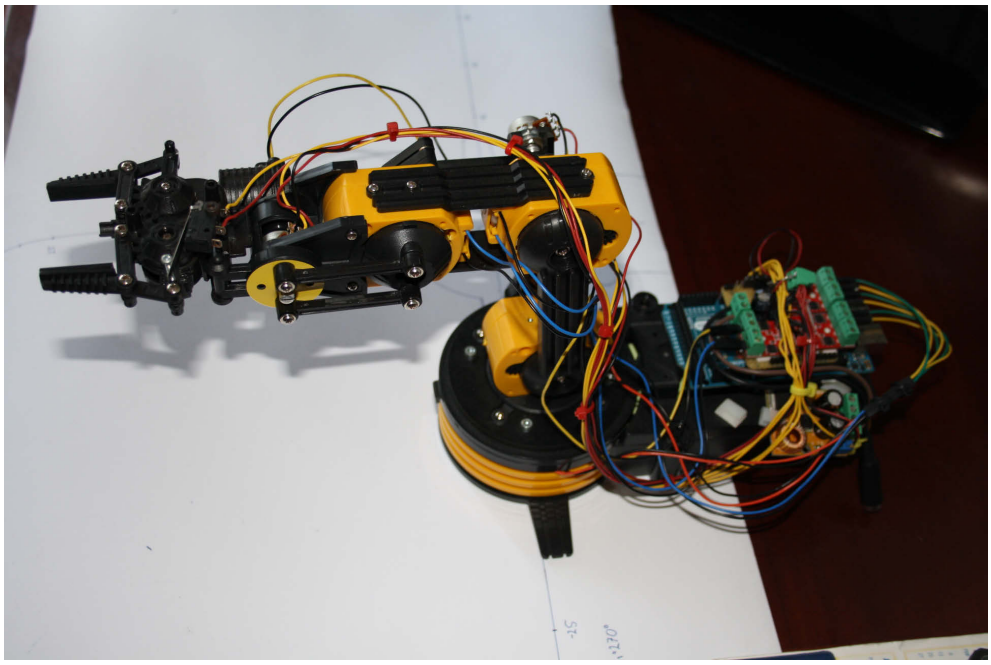
Robotska roka OWI-535 je nizkocenovna robotska roka, ki ima štiri rotacijske sklepe za premikanje in prijemalo (angl. gripper). Glede na način premikanja nekoliko spominja na človeško roko, z razliko, da je ta robotska roka precej bolj okorna oziroma omejena pri premikanju.

Za rešitev izračuna IK za robotsko roko OWI-535 smo uporabili svojevrsten algoritem. Izračun lahko poenostavljeno razdelimo na štiri korake (Slika 1.5), in sicer:

Korak 1: Podamo koordinate X, Y, Z, kamor želimo, da se robotska roka premakne.

Korak 2: Izračun množice rešitev IK (Poglavje 3) je postopek zaporednih matematičnih operacij, ki na podlagi podanih koordinat (točk X, Y, Z) izračuna vrednosti posameznih sklepov robotske roke. Ta postopek se ponovi mnogo krat (odvisno od vrednosti nastavljenih parametrov), zato dobimo v večini primerov tudi veliko število rešitev.

Korak 3: Izbira najboljše rešitve (Poglavje 4) je postopek, ki izbere optimalno rešitev



Slika 1.4: Robotska roka OWI-535.

iz množice rešitev, ki jih dobimo v izračunu IK. Glavni parameter, po katerem se končna rešitev izbere, je razdalja med staro in novo pozicijo posameznega sklepa. Manjša kot je torej ta razdalja, boljša je rešitev.

Korak 4: Dobimo končen rezultat, ki je sestavljen iz kotov θ_1, θ_2, θ_3 in θ_4.

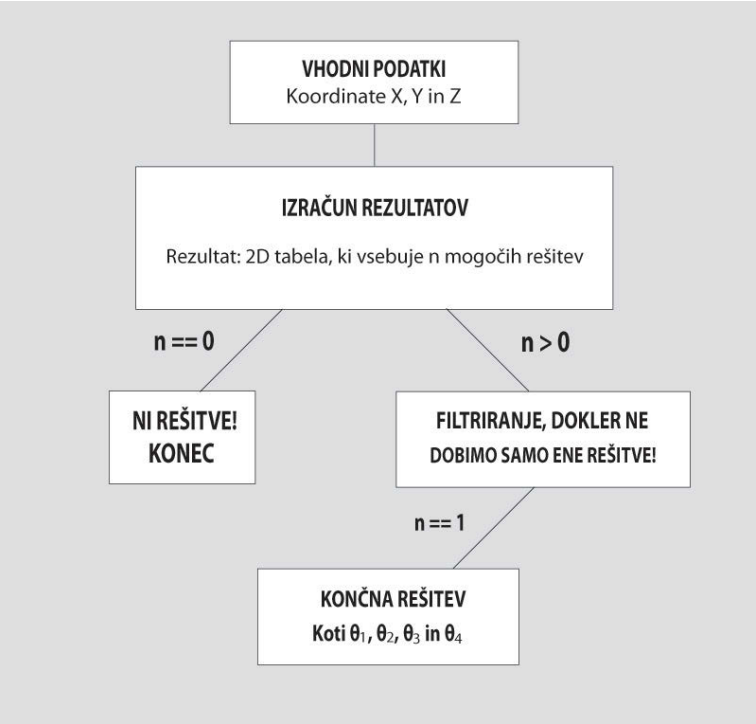
1.3 Zgradba diplomske naloge

Diplomska naloga je okvirno sestavljena iz štirih delov.

V prvem delu (Poglavje 2) je na splošno predstavljena robotika, in sicer glavni pojmi, brez katerih si robotike ni mogoče razlagati. Poleg tega je tu predstavljena tudi robotska roka, za katero smo računali inverzno kinematiko in na njej naredili eksperimentalne meritve. Sledi še opis programske opreme, ki smo jo uporabljali, in kako sta strojna in programska oprema povezani med sabo.

V drugem delu (Poglavje 3) je opisan izračun IK. Tu so podrobno opisani algoritmi, s katerimi se izračunajo vrednosti za vsakega izmed štirih sklepov. V tem poglavju še ne dobimo končnega rezultata IK, ampak množico rezultatov.

Tretji del vsebuje (Poglavje 4) opis postopkov izbire končnega rezultata, ki ga dobimo iz množice, pridobljene v drugem delu. Najprej pretvorimo izračunane vrednosti v drugačen format, ki ga robotska roka lahko prejme, saj je format, ki ga dobimo pri



Slika 1.5: Grafični prikaz poteka izračuna inverzne kinematike v nekaj korakih, kjer n pomeni število rešitev.

izračunu, neprimeren. Nato izmed vseh podatkov algoritem izbere najbolj primernega. Najbolj primeren rezultat je v našem primeru tisti, ki omogoča robotski roki čim hitrejši premik na podano točko. To pa se izračuna glede na razliko med trenutno pozicijo in ciljno pozicijo posameznega sklepa. Opcijsko se lahko vklopi možnost, da roka pride na ciljno točko pod določenim kotom glede na horizontalno ravnino in tako npr. omogočimo lažji prijem predmeta.

V četrtem delu (Poglavje 5) so predstavljeni rezultati meritev iz katerih je razvidno, kako natančna je robotska roka OWI-535 pri premikanju iz ene točke v drugo.

Poglavje 2

Strojna oprema

2.1 O robotiki

Za lažje razumevanje diplomske naloge in robotike na splošno, je v tem poglavju opisanih nekaj osnovnih pojmov, ki se uporabljajo v robotiki.

2.1.1 Osnovni pojmi

Robotska roka je skupek sklepov (zglobov), ki so povezani s togimi elementi, na koncu zadnjega elementa pa se nahaja prijemalo ali drug pripomoček za premikanje ali obdelavo.

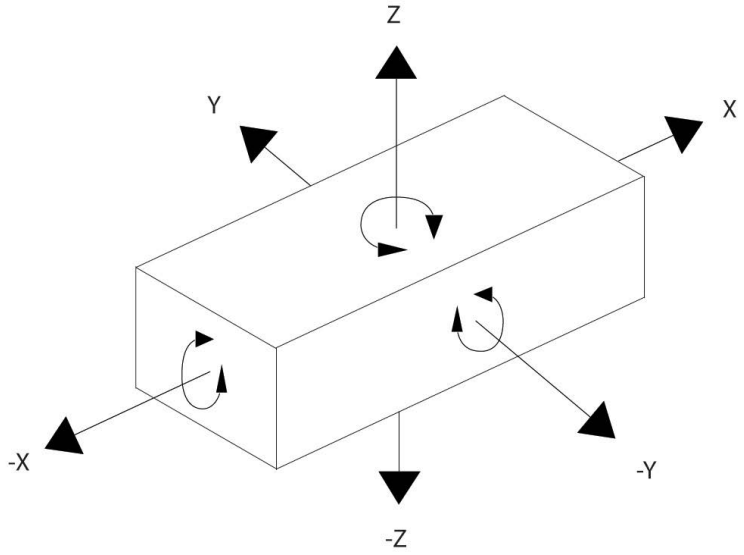
Prostostna stopnja (DOF - Degrees Of Freedom) v robotiki pomeni število spremenljivk potrebnih za opis lege robotskega manipulatorja v prostoru. Da lahko neki predmet opišemo v prostoru, potrebujemo 6 prostostnih stopenj (Slika 2.1). Tri prostostne stopnje določajo pozicijo predmeta, po tri pa orientacijo predmeta v prostoru. Robotska roka OWI-535 ima štiri prostostne stopnje.

Ločimo dve **vrsti sklepov**, in sicer:

- **Translacijski sklep** omejuje gibanje dveh sosednih segmentov na translacijo. Relativni položaj med segmentoma merimo kot razdaljo vzdolž osi sklepa [11].
- **Rotacijski sklep** omejuje gibanje dveh sosednih segmentov na rotacijo. Relativni položaj med segmentoma merimo kot kot zasuka okrog osi sklepa [11].

Glede na **zgradbo** ločimo robotske roke v šest glavnih skupin:

- **Kartezična robotska roka** (angl. cartesian robot arm): Ima tri translacijske sklepe, vsakega za eno os. Delovno območje ima v obliki pravokotnika in je pogosto uporabljena v industriji.



Slika 2.1: Opis predmeta v prostoru s 6 prostostnimi stopnjami.

- **Valjna (cilindrična) robotska roka** (angl. cylindrical robot arm): Je robotska roka, ki je mešanica rotacijskih in translacijskih sklepov. Tipična cilindrična robotska roka ima en rotacijski sklep (primarni sklep) in dva translacijska sklepa. Roka ima delovno območje v obliki valja.

- **Krogelna (sferična) robotska roka** (angl. spherical robot arm): Ima dva rotacijska sklepa in en translacijski sklep.

- **SCARA** (angl. Selective Compliance Assembly Robot Arm) **robotska roka:** Za to vrsto robotskih rok je značilno, da imajo več rotacijskih sklepov, os katerih leži v vertikalni ravnini.

- **Artikulirana (zgibna) robotska roka** (angl. articulated robot arm): Ima samo rotacijske sklepe. Lahko je preprosta in ima zgolj dva sklepa, kompleksnejše roke take vrste pa jih imajo tudi po pet ali več.

- **Paralelna robotska roka** (angl. parallel robot arm): Je nekoliko drugačna od ostalih robotskih rok. Ima več vzporednih segmentov, ki se stikajo na eno prijemalo, ki ga skupaj kontrolirajo. Take robotske roke imajo veliko moč, so zanesljive in zelo hitre, vendar so omejene na majhno delovno območje.

Delovno območje robotske roke je odvisno od vrste robotske roke. Ima lahko obliko **krogle**, **valja**, **pravokotnika** ipd. Robotska roka OWI-535 ima delovno območje v obliki krogle.

Prijemalo se nahaja na koncu zadnjega elementa in je osnova, da lahko predmet prestavimo iz ene pozicije na drugega. Ločimo več vrst prijemal, kot so klešče (z dvema ali več prsti), vakuum, magnet ipd.

Robotska roka ima lahko na vrhu robota tudi orodje, ki ni namenjeno premikanju, ampak na primer **varjenju**, **rezanju**, **barvanju** ipd.

2.1.2 Geometrijski model robotske roke

Geometrijski model robotske roke (tudi direktna kinematika) nam poda pozicijo in orientacijo robotske roke, ki jo izračunamo na podlagi vrednosti danih sklepov ter dolžine segmentov.

Z Denavit-Hartenbergovimi pravili si lahko zelo poenostavimo računanje geometrijskega modela robotske roke. Da element robotske roke opišemo v prostoru, potrebujemo štiri parametre. Izmed teh so tri konstante, eden pa spremenljivka. Kateri je spremenljivka, je odvisno od tega, ali gre za translacijski ali za rotacijski sklep [11].

2.1.3 Inverzni geometrijski model robotske roke

Ta model deluje nasprotno kot model iz prejšnjega poglavja. Za določeno lego vrha robota je potrebno izračunati vrednosti posameznih sklepov. Običajno je tu na voljo veliko število mogočih rešitev, izmed teh pa je potrebno izbrati za uporabnika čim bolj ustrezno.

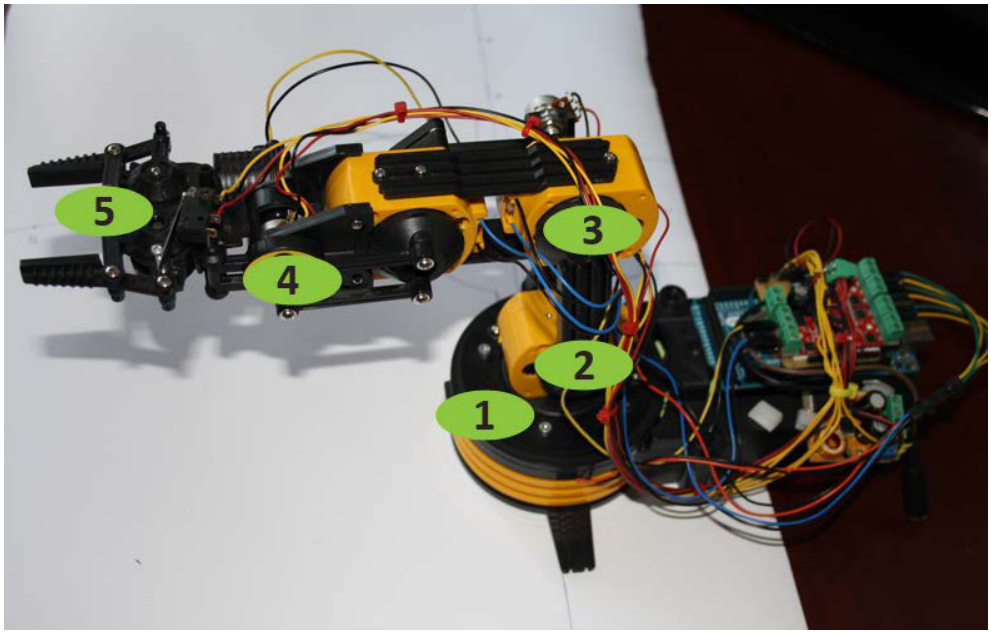
Za računanje IK se pogosto uporablja Jacobijeva matrika (angl. Jacobion matrix), vendar v našem primeru le te ne bomo uporabljali.

2.2 Robotska roka OWI-535

Robotska roka OWI-535 [6] je nizko cenovna robotska roka, ki je sestavljena iz podnožja, štirih sklepov in prijemala, ki so povezani s togimi segmenti (Slika 2.2). V osnovi je premikanje robotske roke mogoče prek žičnega krmilnika. Roka vsebuje pet majhnih elektromotorjev, ki omogočajo premikanje. Vsi elektromotorji za premikanje so enakih moči, kljub temu, da so ob delovanju različno obremenjeni. Označbe posameznih sklepov so vidne na Sliki 2.2. Podrobnejši opis lastnosti sklepov (možnosti rotacije) je naveden v Tabeli 2.1, kjer so navedene vrednosti uradnih specifikacij roke [6]. Ker vrednosti za razpon rotacije iz uradnih specifikacij ne držijo popolnoma, so v isti tabeli (Tabela 2.1) navedene še dejanske vrednosti, pridobljene s pomočjo meritev. V treh primerih je ta

razpon manjši, v enem primeru pa večji. V celotnem izračunu IK smo upoštevali rotacijo, ki je zapisana v uradnih specifikacijah, na koncu pa smo rezultate ustrezno prilagodili.

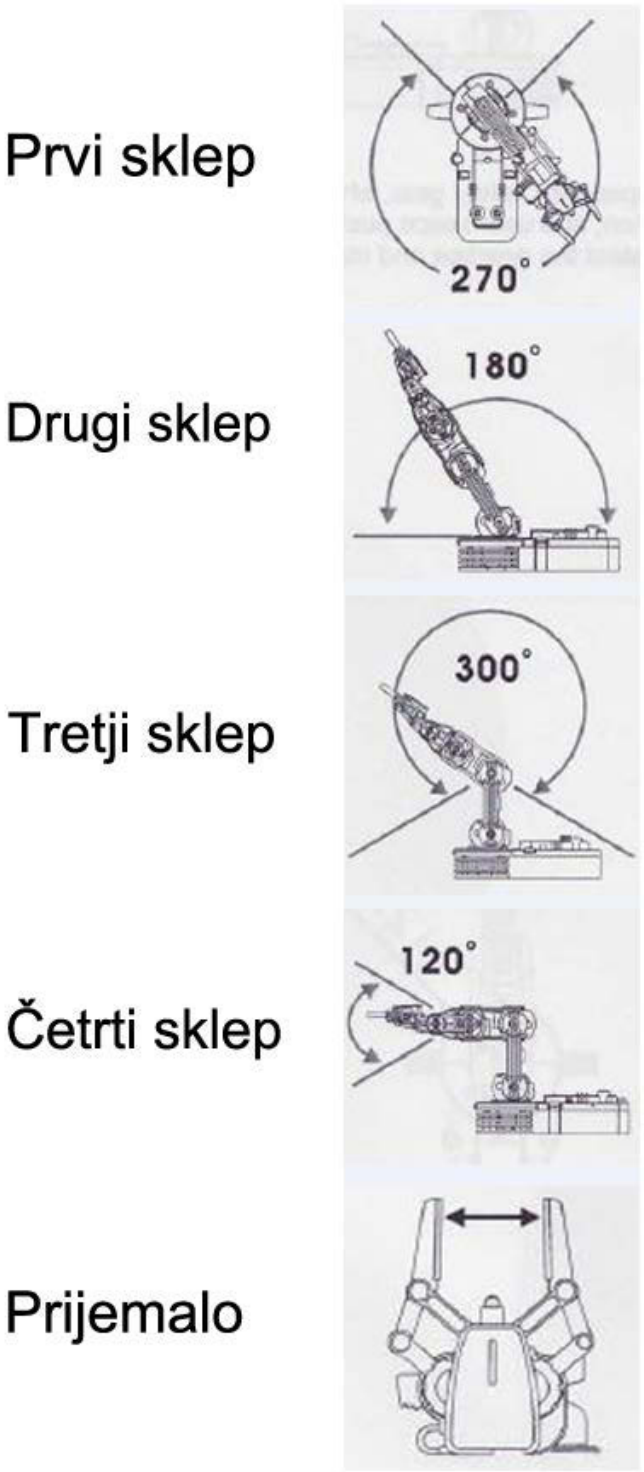
Skupen doseg robotske roke je 27cm, če merimo od prvega sklepa do prijemala, v poziciji, ko je roka popolnoma iztegnjena. V Tabeli 2.2 so navedene dolžine posameznih segmentov robotske roke. Robotska roka OWI-535 je sorazmerno majhna, omogoča pa premikanje lažjih predmetov, ki so težki do 100 gramov [6]. Kljub temu omogoča raznovrstno premikanje v prostoru, kar omogočajo štirje rotacijski sklepi.



Slika 2.2: Robotska roka OWI-535 z označbami posameznih sklepov in prijemala.

	Sklep	Razpon-specifikacije/dejansko (°)	Prijemalo (cm)
1	Prvi sklep	270/250	/
2	Drugi sklep	180/194	/
3	Tretji sklep	300/260	/
4	Četrti sklep	120/100	/
5	Prijemalo	/	4.5

Tabela 2.1: Podrobnejši opis razpona rotacije sklepov in razpona prijemala za robotsko roko OWI-535, označbe se nanašajo na Sliko 2.2.



Slika 2.3: Grafičen prikaz možnosti premikanja posameznih sklepov robota OWI-535. Slika povzeta po [6].

Začetek segmenta	Konec segmenta	Razdalja (cm)
Tla	1	5
1	2	2
2	3	9
3	4	11.3
4	5	6.7

Tabela 2.2: Seznam razdalij med posameznimi sklepi robotske roke OWI-535, označbe se nanašajo na Sliko 2.2.

Prijemalo robotske roke je enostavno in je sestavljeno iz dveh prstov, med katerima je razmik med 0 in največ 4,5cm.

Robotska roka OWI-535, konkretno ta, na kateri so se izračuni tudi testirali, je bila za potrebe lažjega testiranja in programiranja predelana oziroma nadgrajena. Dodani so bili potenciometri, ki vrnejo informacijo, na kateri poziciji se posamezen sklep v danem trenutku nahaja. Pred predelavo ni bilo mogoče vedeti, na katerih pozicijah so se posamezni sklepi robotske roke nahajali v nekem trenutku, kar je pomenilo veliko težavo pri nadzoru. Robotska roka se krmili prek mikrokontrolne plošče Arduino Mega ADK, s pomočjo dodatnega ojačevalnega vezja. Ojačevalno vezje je bilo dodano, da omogoča dovolj visok električni tok za pogonske elektromotorčke. Izhodi na Arduin-u originalno omogočajo tok le do 40mA, kar je bilo za motorčke na roki OWI-535 premalo. Te nadgradnje roke niso bile del te diplomske naloge, več o tem pa je na voljo v tehničnem poročilu [12].

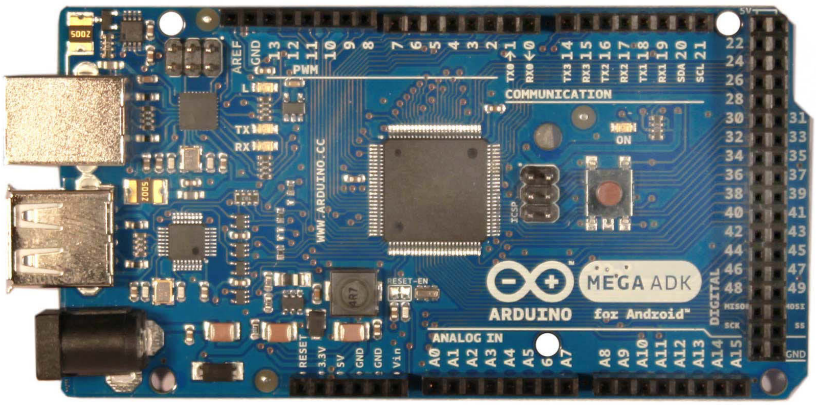
2.3 Arduino

Komunikacija med računalnikom in robotsko roko poteka prek mikrokontrolne plošče Arduino Mega ADK [5] (Slika 2.4).

Mikrokontrolna plošča deluje na frekvenci 16MHz in ima 256KB pomnilnika, napajana pa je z napetostjo 5V. Z računalnikom je povezana prek univerzalnega serijskega vodila (USB - angl. Universal Serial Bus), prek katerega poteka tudi električno napajanje. Ima 54 digitalnih vhodov/izhodov in 16 analognih vhodov. Originalna izhodna moč za izhode, ki jih uporabljamo v našem primeru je 40mA.

2.4 ROS

Krmiljenje robotske roke je realizirano v Robotskem operacijskem sistemu (ROS - Robot Operating System). ROS je odprtokodni sistem, ki vsebuje zbirko knjižnic in orodij, ki



Slika 2.4: Sprednji del mikrokontrolne plošče Arduino Mega ADK. Slika povzeta po [5].

pripomorejo k lažjemu programiranju robotskih naprav. ROS je bil razvit v letu 2007 pod imenom Switchyard v laboratoriju za umetno inteligenco v Stanfordu ob projektu Stanford AI Robot, kasneje pa se je razvoj nadaljeval v raziskovalnem inštitutu Willow Garage. Skozi leta se je nadgrajeval v sodelovanju z več kot 20 institucijami [9, 10].

Razvija se v programskem jeziku C++ in deluje na operacijskem sistemu Linux. ROS omogoča pisanje osnovne programske opreme za robote (npr. gonilniki naprav), kot tudi naprednejše algoritme za nadziranje robotov. Ta diplomska naloga zajema predvsem slednji del.

Koda v ROS-u se piše v posebnih izvršljivih datotekah, ki se imenujejo **vozlišča** (angl. nodes). V našem primeru se celotna koda za izračun inverzne kinematike nahaja v eni taki datoteki, ki je razdeljena na več manjših (pomožnih) funkcij in na glavno funkcijo (angl. main function). Zagonske datoteke se zaženejo prek ukazne lupine (angl. shell). Da lahko te izvršljive datoteke zaženemo je potrebno predčasno zagnati ukaz “roscore”, ki omogoča nekatere osnovne funkcionalnosti vozlišč, kot je npr. komunikacija med le temi.

2.4.1 Struktura delovnega okolja

Delovno okolje (angl. workspace) v ROS-u je mapa, v kateri se nahaja vsa vsebina določenega samostojnega projekta (ang. project). Delovno okolje je sestavljeno iz več prostorov:

- prostor izvorne kode (ang. source space),

- prostor za gradnjo (ang. build space),
- prostor za razvoj (ang. development space),
- prostor za namestitev (ang. install space),
- prostor za rezultate (ang. result space).

2.4.2 Osnovni ROS ukazi

ROS ukaze se izvaja prek ukazne lupine. Nekateri ukazi se lahko izvedejo samostojno, nekaterim pa je potrebno podati tudi vhodne podatke. Določenim ukazom se lahko prek stikal spremeni način privzete izvedbe ukaza. Nekaj osnovnih ROS ukazov:

ROSCORE je zbirka vozlišč in programov, ki so potrebni za vsak ROS sistem. Zažene se z istoimenskim ukazom "roscore", potreben pa ni noben parameter. Ta ukaz nam omogoča osnovne funkcionalnosti sistema ROS.

ROSRUN ukaz omogoča zagon izvršljivih datotek v ROS-u. Zagon datoteke je mogoč znotraj poljubnega paketa (ang. package), brez potrebnega navajanja točne poti izvršljive datoteke.

CATKIN_MAKE je ukaz, ki zgradi (ang. build) kodo v našem delovnem prostoru (catkin workspace). Po vsakem spreminjanju vsebine v izvršljivih datotekah je potrebno izvesti ta ukaz, da so spremembe nato tudi upoštevane.

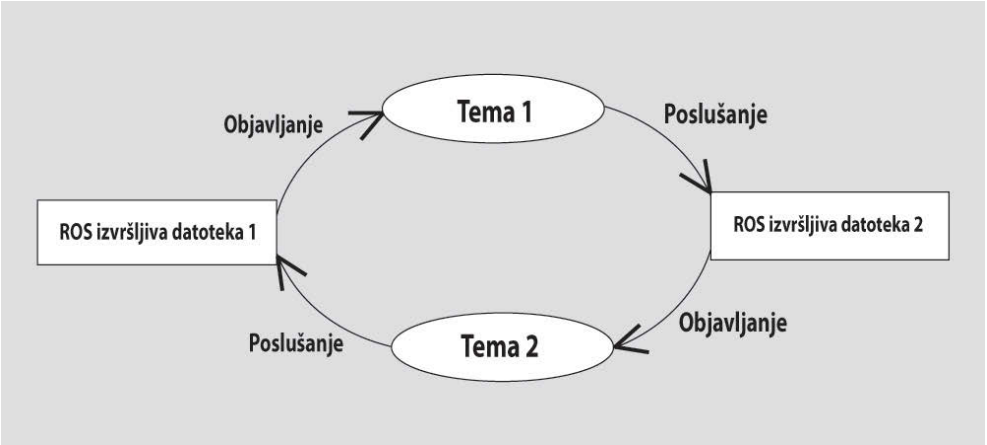
ROSNODE je ukaz, ki omogoča izpis informacij in upravljanje izvršljivih datotek.

ROSTOPIC ukaz omogoča prikaz vsebine tem (ang. topic).

2.4.3 Komunikacija med izvršljivimi datotekami

Če imamo več izvršljivih datotek, je med njimi mogoča komunikacija. Komunikacija poteka prek objavljanja (angl. publish) in naročanja (angl. subscribe) na določene teme. Teme se strogo ločijo glede na vrsto sporočila (angl. message), ki ga lahko posredujejo. Vsaka tema ima točno določeno vrsto sporočila. Če vrsta sporočila ni ustrezna, se povezava med temo in vozliščem ne more vzpostaviti. Za vsako temo je lahko več objavljavcev kot tudi naročnikov.

V našem primeru imamo dve izvršljivi datoteki. Ena je bila napisana že predhodno in ni del diplomske naloge. Le ta nam omogoča premikanje robotske roke, saj ta datoteka komunicira direktno z gonilnikom. Druga izvršljiva datoteka (Poglavje 2.4.4, zadnja alineja) je v celoti del diplomskega dela in vsebuje celoten izračun IK.



Slika 2.5: Graf prikaza komunikacije med ROS izvršljivima datotekama prek tem.

Izvršljivi datoteki komunicirata med sabo obojestransko prek tem. Prva npr. pošilja podatke drugi datoteki o trenutnih vrednostih kotov posameznih sklepov. Druga pa pošilja podatke prvi datoteki, kam naj se posamezen sklep robotske roke premakne (rezultat izračuna IK). To sta dva najbolj pogosta načina komunikacije.

2.4.4 Zagon izvršljive datoteke za premik z IK

Na računalniku je potrebno imeti naložen operacijski sistem Linux, na njem pa ROS. V delovni prostor (v našem primeru je to catkin) naložimo paket, v katerem se nahaja tudi izvršljiva datoteka, ki je del diplomske naloge.

Vse ukaze izvajamo prek ukazne lupine, moramo pa biti v mapi svojega delovnega prostora. Ukazi (opisani v Poglavju 2.4.2), ki jih moramo nato vnesti v takem zaporedju, kot so našteti tukaj, so:

- roscore,
- catkin_make,
- rosrun small_arm_node,
- rosrun move_arm_node.

Ko vnesemo zadnji ukaz (to je ukaz za premik roke), je v ukazno lupino potrebno vnesti koordinate. Za tem še določimo, ali vključimo način za prijem objekta ali ne (Poglavje 4.5). Ko vnesemo te podatke se, na podlagi izračunov IK roka premakne na podano točko, ali pa vrne napako, če ni nobene rešitve oziroma so vhodni podatki narobe podani.

Na Sliki 2.6 se lahko vidi izpis ukazne lupine pred in med premikanjem robotske roke. Vidi se izpis koordinat, kam se bo roka premaknila (prva vrstica), izpis kotov sklepov v trenutku izpisa in kam se morajo ti sklepi premakniti (od druge do devete vrstice) ter pozicije sklepov med premikanjem robotske roke (od desete vrstice dalje). Izpis podatkov trenutne pozicije se izvede pri vsakem desetem obhodu zanke.

```
Move to coordinates: X: -3.000000, Y: 21.000000, Z: -6.000000, x2d: 6.708204
Motor 1 (current): 126
Move   motor   1   to:    9.517400
Motor 2 (current): 100
Move   motor   2   to:    47.000000
Motor 3 (current): 245
Move   motor   3   to:    237.942322
Motor 4 (current): 84
Move   motor   4   to:    118.094734
Positions: M1: 126, M2: 100, M3: 245, M4: 84, M5: 0
Positions: M1: 126, M2: 99, M3: 241, M4: 90, M5: 0
Positions: M1: 116, M2: 84, M3: 235, M4: 104, M5: 0
Positions: M1: 106, M2: 71, M3: 235, M4: 117, M5: 0
Positions: M1: 94, M2: 61, M3: 235, M4: 119, M5: 0
Positions: M1: 88, M2: 48, M3: 235, M4: 119, M5: 0
Positions: M1: 78, M2: 48, M3: 235, M4: 119, M5: 0
Positions: M1: 68, M2: 48, M3: 235, M4: 119, M5: 0
Positions: M1: 57, M2: 48, M3: 235, M4: 119, M5: 0
Positions: M1: 47, M2: 48, M3: 235, M4: 119, M5: 0
Positions: M1: 37, M2: 48, M3: 235, M4: 119, M5: 0
Positions: M1: 28, M2: 48, M3: 235, M4: 119, M5: 0
Positions: M1: 19, M2: 48, M3: 235, M4: 119, M5: 0
Positions: M1: 13, M2: 48, M3: 235, M4: 119, M5: 0
Positions: M1: 10, M2: 48, M3: 235, M4: 119, M5: 0
Positions: M1: 10, M2: 48, M3: 235, M4: 119, M5: 0
```

Slika 2.6: Izpis ukazne lupine tik pred začetkom premikanja do konca premika.

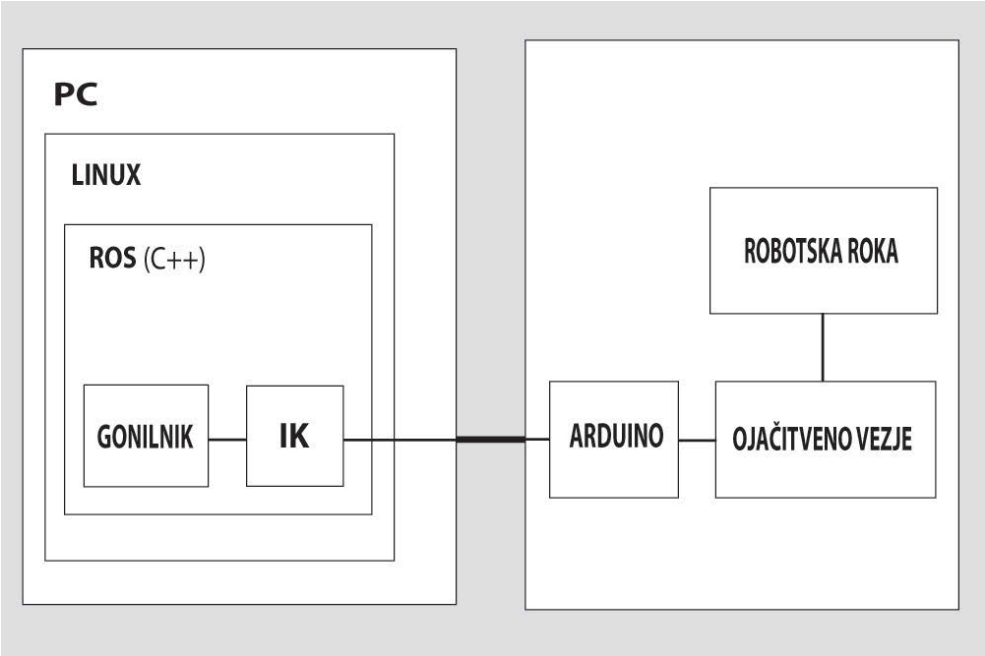
2.4.5 C++

Celotna programska koda za izračun IK je napisana v programskem jeziku C++. C++ je splošno namenski programski jezik, ki ga je razvil danski računalniški strokovnjak Bjarne Stroustrup leta 1983. Skozi leta se je zelo spreminjal. Dodajali so mu funkcionalnosti, kot so razredi, dedovanje, virtualne funkcije, preobložitev operatorjev, rokovanje z izjemami in predloge. Od devetdesetih let prejšnjega stoletja naprej je med najbolj uporabljenimi programskimi jeziki na svetu [7, 8].

2.5 Povezava naprav

Vsi elementi iz zgornjih poglavij sestavljajo povezan sistem naprav (Slika 2.7). Na osebnem računalniku, na katerem je naložen operacijski sistem Linux, imamo naloženo ogrodje ROS. V tem ogrodju programiramo ROS izvršljive datoteke. Preko gonilnika, ki smo ga

imeli že predhodno napisanega, se podatki pošiljajo na kontrolno ploščo Arduino, ki je prek dodatnih ojačevalnih vezij povezana z robotsko roko.



Slika 2.7: Skica povezav od računalnika do robotske roke.

Poglavje 3

Izračun inverzne kinematike

Cilj izračuna IK je podatke, ki jih algoritem sprejme, to je pozicija točke v 3D prostoru (koordinate X, Y, Z), pretvori v vrednosti sklepov robotske roke. Pri izračunu IK izračunamo le pozicijo brez orientacije, ki pa jo izračunamo v naslednjem poglavju na podlagi podatkov, pridobljenih v tem poglavju.

3.1 Podatki

Vhodni parametri funkcije za izračun IK so koordinate X, Y in Z, ki opisujejo točko v 3D koordinatnem sistemu.

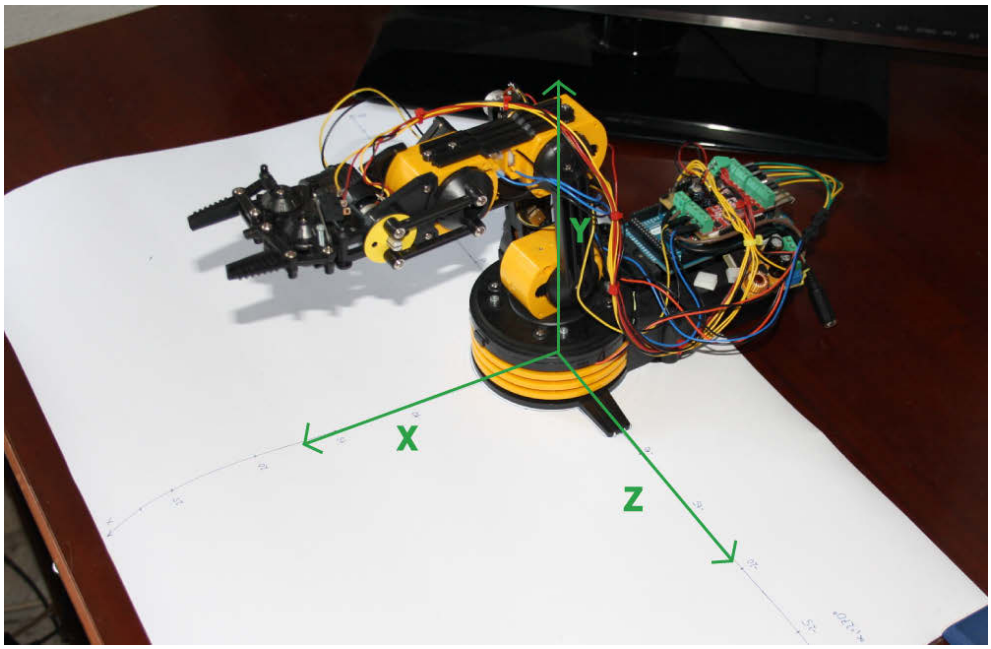
Izhodne vrednosti so vrednosti premika posameznega sklepa (θ_1, θ_2, θ_3 in θ_4). Te vrednosti se podajo posameznim sklepom, ki se nato premaknejo na to mesto.

Konstante definiramo na začetku datoteke, pred začetkom izračunavanja. Konstante, ki jih uporabljamo za izračunom, so:

- dolžine posameznih segmentov,
- maksimalen razpon premika vsakega sklepa.

V postopku izračunavanja IK je veliko število začasnih lokalnih in globalnih spremenljivk, tabel, matematičnih konstant ipd.

Robotsko roko postavimo v koordinatni sistem, kjer sta X in Z koordinati, ki opisujeta spodnjo ravnino (podlago), Y koordinata pa opisuje navpično os oziroma višino (Slika 3.1).



Slika 3.1: Postavitev robotske roke v koordinatni sistem, kjer X in Z koordinati opisujeta podlago, Y koordinata pa višino.

3.2 Vračanje vrednosti

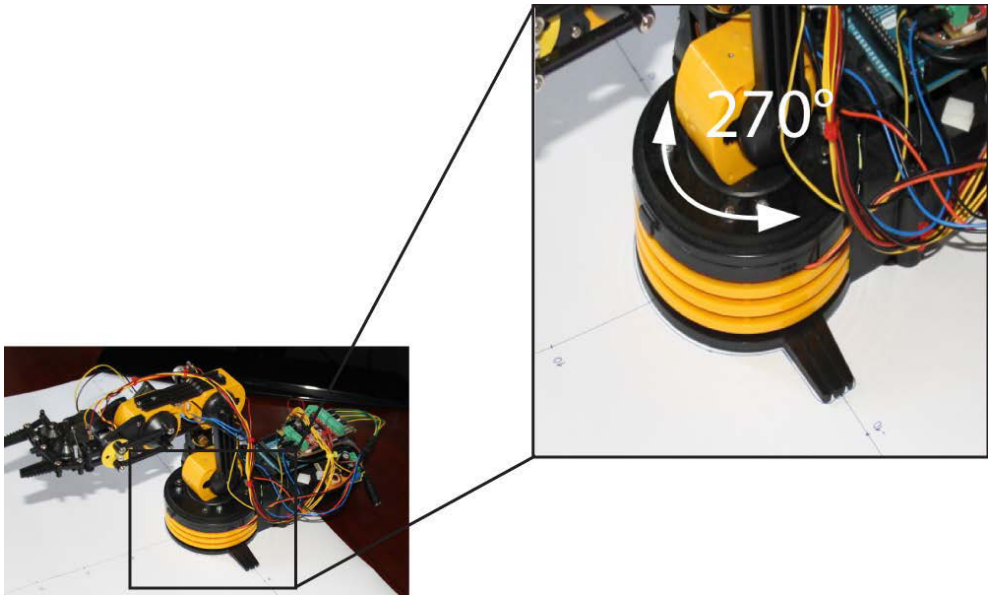
Funkcija za izračun IK v nekaterih primerih vrne obvestilo o napaki v postopku. V ukazni lupini se izpiše ustrezno obvestilo, program pa se samodejno zaključi. To je največkrat v primeru, ko se točka, ki jo iščemo, nahaja na neprimernem mestu. Nekaj primerov, ko funkcija vrne napako:

- če se podana točka nahaja izven dosega roke – točka je oddaljena več kot 27cm od izhodišča robotske roke,
- če se podana točka nahaja znotraj ali preblizu območja, kjer je nameščeno kontrolno vezje,
- če se podana točka nahaja nižje od tal,
- če ni najdena nobena ustrezna rešitev za podano točko.

3.3 Prvi sklep

Prvi sklep (Slika 3.2) se od ostalih razlikuje, saj kot edini rotira okrog navpične osi. To posledično pomeni, da sta od pozicije primarnega sklepa odvisni koordinati Z in X.

Omogoča rotacijo od 0° do 270°. Območje ostalih 90° pa obsega mrtvi kot. Kljub temu roka lahko dostopa do območja teh 90°, saj imajo ostali sklepi dovolj velik razpon (Tabela 4.1), da dostopajo do točke, ki se nahaja na nasprotni strani roke, glede na to, kam je obrnjena.



Slika 3.2: Prvi sklep robotske roke OWI-535.

Pri izračunu vrednosti prvega sklepa v nekaterih primerih obstaja ena rešitev (Slika 3.3, oranžno območje), v ostalih primerih pa dve rešitvi (Slika 3.3, belo območje).

Pri **prvi rešitvi** se prvi sklep preprosto samo premakne iz stare na novo pozicijo.

Druga rešitev pa je definirana tako, da se prvi sklep obrne za 180° glede na prvo rešitev, ostali sklepi pa se premaknejo tako, da je vrh robota na enaki poziciji, kot pred premikom prvega sklepa.

Različno število rešitev imamo, ker ima prvi sklep možnost rotacije za 270°. Če bi prvi sklep omogočal rotacijo za 360°, bi v celotnem območju obstajali dve mogoči rešitvi, če pa bi omogočal rotacijo za 180°, bi bila za prvi sklep vedno samo ena rešitev. Rešitev je neodvisna od vrednosti ostalih treh sklepov. Glede na koordinati X in Z izračunamo pozicijo prvega sklepa po naslednjih dveh korakih:

Korak 1: ta korak je potreben ne glede na to, kje se točka nahaja. Sestavljen je iz dveh delov:

- Izračunamo razdaljo (diagonalo) D med koordinatama X in Z:

$$D = \sqrt{X^2 + Z^2} \tag{3.1}$$

- Izračunamo kot θ_1, ki nam pove, kolikšna je vrednost prvega sklepa:

$$\theta_1 = acos\left(\frac{X}{D}\right) \tag{3.2}$$

Korak 2: ta korak se uporabi, v kolikor se točka, kamor se roka premakne, nahaja v belem območju (Slika 3.3). Postopek je enak kot pri prvem koraku, z dodatkom na koncu. Kot se spremeni za 180° in tako dobimo še drugo rešitev. Ta korak je sestavljen iz treh delov. V prvih dveh delih sta rezultat D in θ_1, enako, kot v prvem koraku, sledi pa še tretji del:

- Izračunamo obratno vrednost θ_1' po navedeni formuli in dobimo rešitev koraka 2:

$$\theta_1' = abs(\theta_1 - 180) \tag{3.3}$$

Primeri glede na število rešitev:

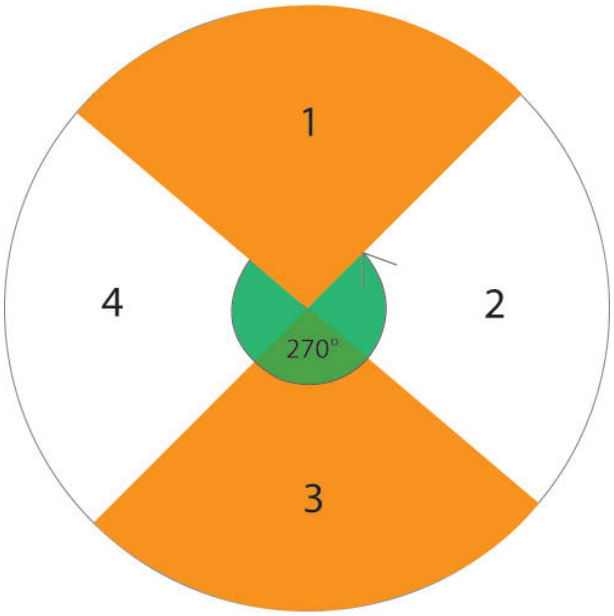
- Če se točka, za katero računamo IK, nahaja v območju 90-ih stopinj, kamor roka direktno nima dostopa (območje 1 na Sliki 3.3), je mogoča samo ena rešitev.
- Če se točka nahaja na nasprotni strani zgoraj omenjenega območja (območje 3 na Sliki 3.3), je prav tako mogoča samo ena rešitev.
- V vseh drugih primerih, torej na območjih obarvanih z belo (Slika 3.3), sta mogoči 2 rešitvi.

Za prvi sklep v območju dosega robotske roke sta za polovico vseh obstoječih točk mogoči dve rešitvi. To sta beli območji na Sliki 3.3, za drugo polovico točk pa ena rešitev, kar je označeno z oranžno barvo na Sliki 3.3. Če robotska roka ne bi imela nikakršnih omejitev glede premikanja (kot je npr. dodatna omejitev v polju 1, kjer se nahaja krmilna plošča Arduino), potem bi lahko trdili, da ima natanko polovico točk eno možnost dostopa ter polovico točk dve možnosti dostopa za prvi sklep.

Pri drugi rešitvi je postavitev sklepov taka, da je roka obrnjena ravno obratno. Del robotske roke, ki je običajno **na vrhu,** je v tem primeru **spodaj**. Del roke, ki je običajno **spodaj**, pa je v tem primeru na **vrhu**.

Na območju, označenem s številko 3 (Slika 3.3) je robotska roka vedno obrnjena pravilno, na območju 1 (Slika 3.3) pa stalno obratno.

Če sta dve mogoči rešitvi, je potrebno izbrati bolj primerno. Več o tem, katero možnost izbere naš algoritem, je opisano v Poglavju 4.4.2.



Slika 3.3: Prikaz območja gibanja glede na število možnih rešitev za prvi sklep.

3.4 Poenostavitev koordinatnega sistema

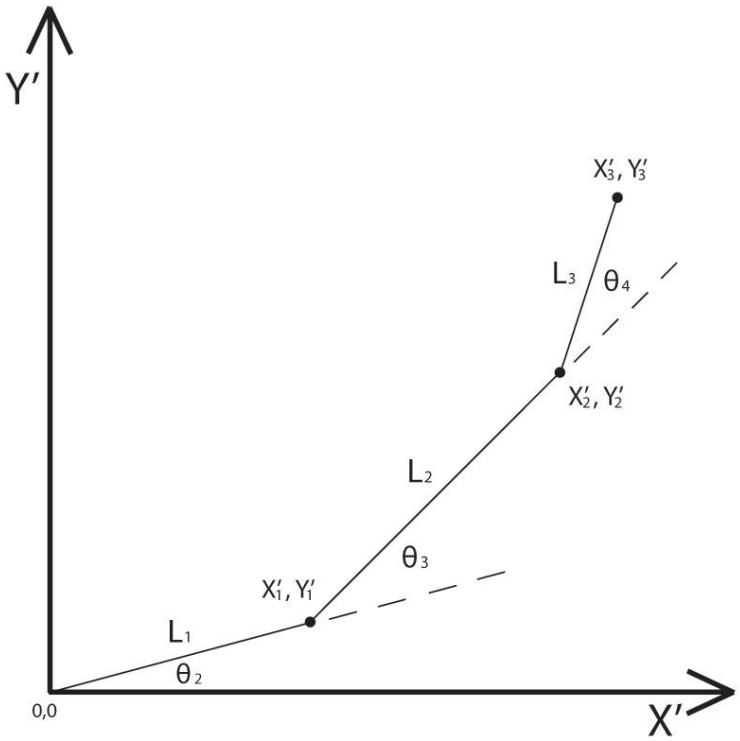
Premikanje robotske roke poteka v 3D prostoru. Prvi sklep nima neposredne povezave z ostalimi tremi sklepi za premikanje pri izračunu IK. Zato lahko pri izračunu IK prvi sklep izpustimo in gledamo samo ostale tri sklepe. Tako si je lažje predstavljati s kakšno težavo se soočamo. Če izločimo prvi sklep, lahko premikanje iz 3D prostora, prestavimo na 2D ravnino.

Zdaj imamo sistem treh rotacijskih sklepov in treh togih segmentov na 2D ravnini (Slika 3.4). Koordinati, s katerima je koordinatni sistem označen, sta X' in Y', kjer je X' vrednost, ki nam pove, kolikšna je oddaljenost od izhodišča na spodnji ravnini (podlagi), natanko pod vrhom roke, Y' pa je višina, torej oddaljenost od vrha roke do podlage, na kateri stoji robotska roka. Os Y' sovpada z osjo Y v izhodiščnem koordinatnem sistemu, os X' pa je za kot θ_1 zasukana os okrog osi Y. Vse spremenljivke v naslednjih dveh poglavjih se nanašajo na Sliko 3.4.

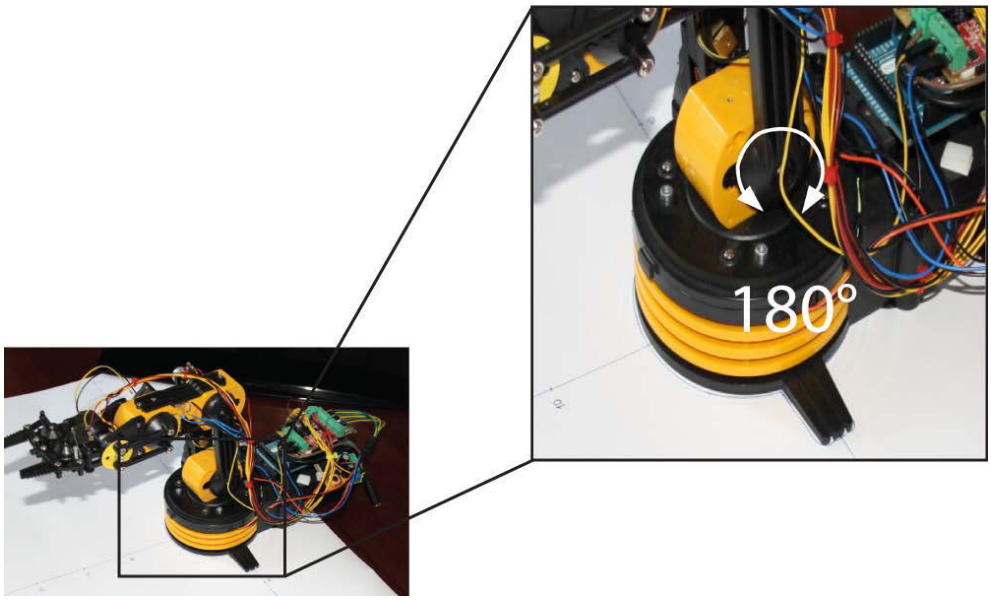
3.5 Izračun kotov za drugi sklep

Kot drugega sklepa (θ_2) ima razpon med 0 in 180° (Slika 3.5). Ta kot se ne računa, ampak se določijo mogoče vrednosti, ki so enakomerno razporejene med najmanjšo in največjo vrednostjo celotnega razpona sklepa.

Vrednosti sklepa se določijo tako, da gre program z zanko (angl. loop) od začetne do



Slika 3.4: Prikaz segmentov in sklepov v 2D koordinatnem sistemu, kakršnega imamo za izračun IK.

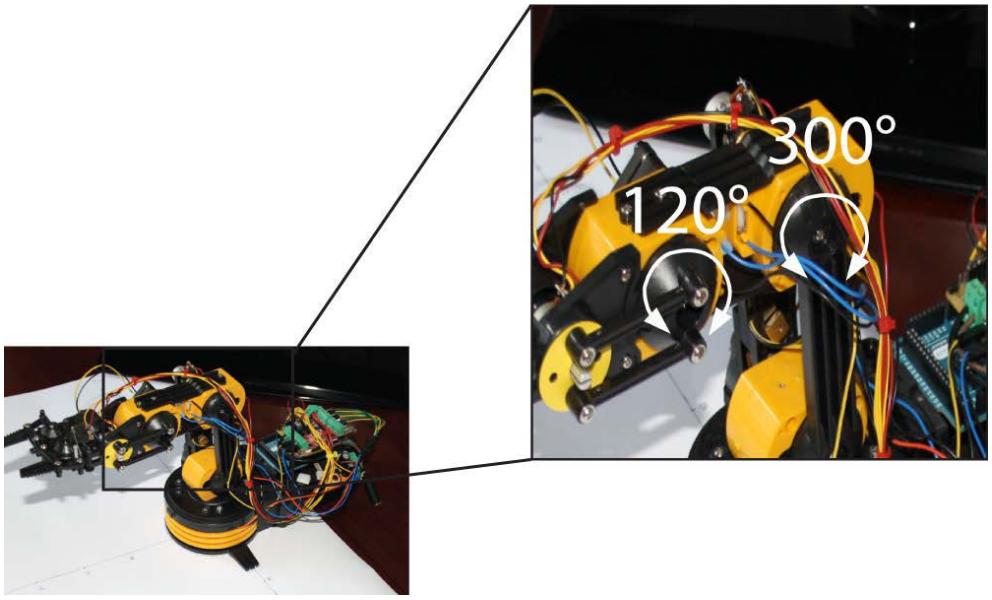


Slika 3.5: Prvi sklep robotske roke OWI-535.

končne vrednosti sklepa z določenim korakom. V našem primeru določimo, da je vrednost koraka pet, ta pa se lahko poljubno spreminja. Če je korak večji, dobimo manj mogočih rešitev, končna rešitev pa je manj natančna. Če je korak manjši, je ravno obratno. Ker je korak enak pet, to pomeni, da dobimo 180/5 obhodov zanke oziroma mogočih vrednosti. Torej je vseh mogočih rešitev v primeru, da ima korak vrednost 5, enako 36.

3.6 Izračun kotov za tretji in četrti sklep

Tretji sklep (Slika 3.6) ima možnost gibanja med 0 in 300°, kar je največ med vsemi sklepi. Četrti sklep (Slika 3.6) pa se lahko giblje med 0 in 120°.



Slika 3.6: Drugi in tretji sklep pri robotski roki OWI-535.

Na podlagi končne točke X_1', $Y_{1'}$ in dolžine segmenta L_1, lahko izračunamo kota θ_3 in θ_4 (Slika 3.7), po postopku, ki je naveden v spodnjih korakih. Vsi koti in točke se nanašajo na Sliko 3.4. Točke X, Y in Z pa so točke, za katere iščemo rešitev IK.

Korak 1:Y_3' je enak koordinati Y, ki ga prejmemo kot vhodni parameter, X_3' pa je enak diagonali med vhodnima vrednostima X in Z:

$$X_3' = \sqrt{X^2 + Z^2} \tag{3.4}$$

Korak 2: Izračunamo poziciji X_1' in Y_1':

$$X_1' = L_1 cos(\theta_2) \tag{3.5}$$

$$Y_1' = L_1 sin(\theta_2) \tag{3.6}$$

Korak 3: Izračunamo končno pozicijo X_3'' in Y_3'' (Slika 3.7), če prestavimo izhodišče koordinatnega sistema v poziciji X_1' in Y_1':

$$X_3'' = X_3' - X_1' \tag{3.7}$$

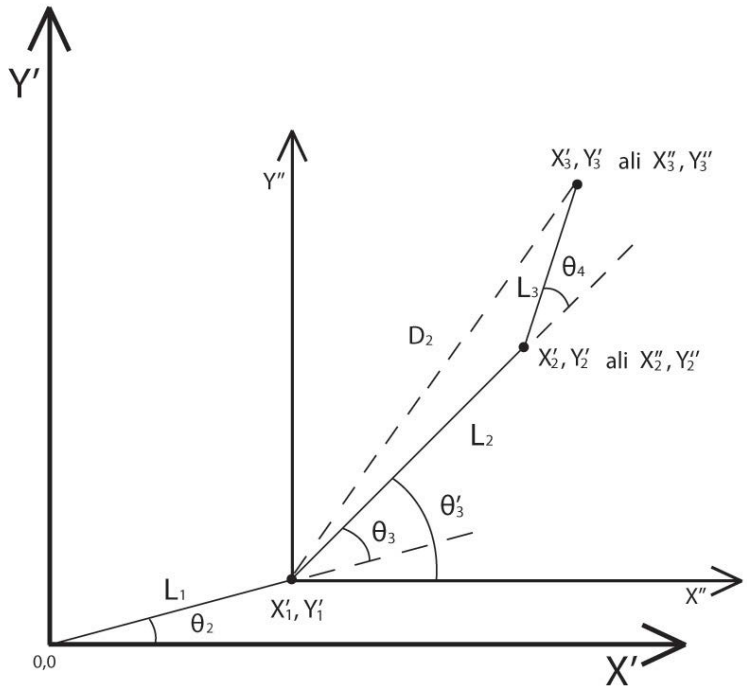
$$Y_3'' = Y_3' - Y_1' \tag{3.8}$$

Korak 4: Izračunamo kot θ_4:

$$\theta_4 = acos\left(\frac{(((X_3'')(X_3'')) + ((Y_3'')(Y_3'')) - (L_2^2) - (L_3^2))}{2L_2L_3}\right) \tag{3.9}$$

Korak 5: S pomočjo enačbe z razmerji kotov izračunamo kot, od katerega odštejemo vrednost kota θ_2 in dobimo kot θ_3:

$$\theta_3 = \left(2\left(atan\left(\frac{Y_3}{X_3 + D_2}\right)\right) - \theta_3'\right) - \theta_2 \tag{3.10}$$



Slika 3.7: Prikaz segmentov in sklepov v 2D koordinatnem sistemu z izhodiščem v izhodišču drugega sklepa in z izhodiščem v točki tretjega sklepa.

V vsakem obhodu zanke (Poglavje 3.5) dobimo eno rešitev, ali nobene, če robotska roka v določeni poziciji nima podane točke v svojem dosegu. V kolikor je končna točka X_3', Y_3' preveč oddaljena ali pa je preblizu končne točke segmenta L_1 (X_1', Y_1'), rešitev v trenutnem obhodu zanke ne obstaja. Ob koncu tega koraka je trenutna rešitev shranjena v obliki 2D tabele (Tabela 3.1).

Po tem koraku je velika verjetnost, da je število mogočih rešitev zelo veliko. To je odvisno tudi od same pozicije točke, za katero iščemo rešitev. Kako dobimo rešitev iz trenutne množice rešitev pa v Poglavju 4.

1.	θ_{1_1}	θ_{2_1}	θ_{3_1}	θ_{4_1}
2.	θ_{1_2}	θ_{2_2}	θ_{3_2}	θ_{4_2}
3.	θ_{1_3}	θ_{2_3}	θ_{3_3}	θ_{4_3}
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
n.	θ_{1_n}	θ_{2_n}	θ_{3_n}	θ_{4_n}

Tabela 3.1: Primer rešitve, ki jo dobimo ob koncu tega koraka.

3.7 Preverjanje pravilnosti rešitev

Ali so rešitve pravilne, se preverja prek funkcije, ki kot vhodne podatke prejme kote treh sklepov (brez prvega sklepa), vrne pa koordinati v prostoru. Prvi sklep se tukaj ne upošteva, saj je rešitev le te dokaj trivialna, funkcija pa je zato lahko preprostejša. Funkcija je torej izračun direktne kinematike za našo robotsko roko. Vrednosti, ki jih vrne funkcija, so višina, koliko je prijemalo oddaljeno od podlage in oddaljenost od izhodišča robotske roke do točke, ki se nahaja na podlagi, natanko pod prijemalom.

Funkcija za preverjanje pravilnosti rešitev se uporablja predvsem za testiranje rezultatov med programiranjem kode za računanje IK. S pomočjo te funkcije se lahko hitro ugotovi, ali je prišlo do računske napake.

Funkcija prejme kote θ_2, θ_3 in θ_4 in dolžine segmentov L_1, L_2 in L_3, ki so konstante. Vrne pa vrednosti X_3' in Y_3' (Slika 3.4):

$$X_3' = L_1 \cos\left(\frac{\theta_2 \pi}{180}\right) + L_2 \cos\left(\frac{\theta_3 \pi}{180}\right) + L_3 \cos\left(\frac{\theta_4 \pi}{180}\right) \tag{3.11}$$

$$Y_3' = L_1 \sin\left(\frac{\theta_2 \pi}{180}\right) + L_2 \sin\left(\frac{\theta_3 \pi}{180}\right) + L_3 \sin\left(\frac{\theta_4 \pi}{180}\right) \tag{3.12}$$

Poglavje 4

Izbor najboljše rešitve

Izbor najboljše rešitve se izvede v primeru, da je množica rešitev iz Poglavja 3 večja od ena. Težava izbora rešitve je, kako **izbrati čim bolj optimalno končno rešitev** iz množice rešitev.

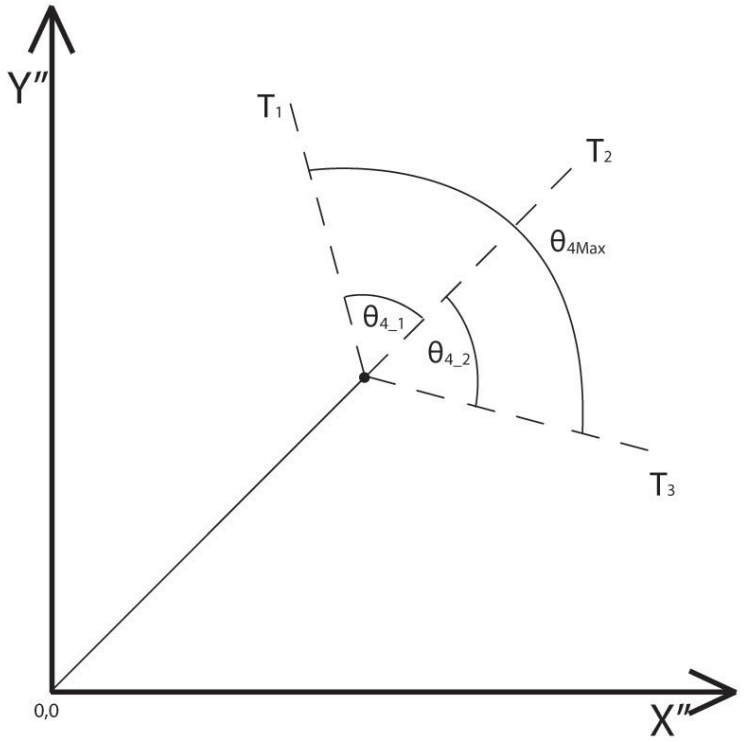
Iz množice rešitev najprej odstranimo neuporabne vrednosti. Izločimo vse rešitve, katere za robotsko roko niso sprejemljive zaradi samih vrednosti kotov pri katerem koli izmed sklepov. Dobljene vrednosti nato pretvorimo v vrednosti, ki jih bomo podali robotu, saj je potrebno spremeniti format podatkov. Po koncu tega koraka ostane še množica rešitev, ki so ustrezne za premik robotske roke. Sledi izločanje manj ustreznih rešitev, dokler ne dobimo najbolj primerne rešitve, ki predstavlja končno rešitev. V našem primeru bo najbolj primerna rešitev, ki bo omogočala čim hitrejši premik iz ene na drugo točko.

4.1 Izločitev neustreznih rešitev

Vsak sklep ima določeno območje, v katerem lahko deluje (Tabela 2.1). Dobljeni rezultati lahko odstopajo od zmožnosti premikanja sklepa robota. Npr. četrti sklep ima za možnost premikanja na območju kota θ_{4Max}, ki je enako 120° (Slika 4.1), torej tega sklepa ne moremo premakniti za več kot 120°, ne glede na to kje se nahaja v danem trenutku. Če bi dobili rešitev, torej vrednost, kam naj se premakne četrti sklep, ki ni v območju kota θ_{4Max}, to rešitev zavržemo.

Obliko vhodnih vrednosti lahko pogledamo na primeru četrtega sklepa, ki ima možnost premikanja na območju do največ 120°, kar je območje θ_{4Max} na Sliki 4.1.

Če robotska roka prejme podatek, da se sklep zapestja premakne na 60°, se le ta premakne na linijo, označeno s T_1. Če robotska roka prejme podatek za premik na vrednosti 0°, pomeni, da se mora zapestje premakniti na linijo T_2, v kolikor pa prejme



Slika 4.1: Podrobnejši prikaz četrtega sklepa.

podatek za premik na -60°, se sklep premakne na črto T_3.

Odstranimo vse rešitve, ki odstopajo od vrednosti, ki jih lahko prejme posamezen sklep. Vsi podatki morajo biti v obliki, kot je opisano v zgornjem odstavku. Zaradi tega razdelimo vsak sklep na dva dela, kot sta θ_{4_1} in θ_{4_2} na Sliki 4.1. Vse rezultate, ki ne ustrezajo veljavnim vrednostim v Tabeli 4.1, odstranimo iz tabele mogočih končnih rešitev.

Število sklepa	Razpon premika (°)	Veljavne vrednosti (min-max)
Prvi sklep	270	-135 do 135
Drugi sklep	180	-90 do 90
Tretji sklep	300	-150 do 150
Četrti sklep	120	-60 do 60

Tabela 4.1: Veljavne vrednosti sklepov v formatu po izračunu IK.

4.2 Določanje parametrov rotacije

Podatki, ki jih robotska roka (potenciometri na robotski roki) lahko prejme, morajo biti zapisani v vrednosti med 0 in θ_{Max}, kjer je θ_{Max} največja vrednost posameznega sklepa

oz. razpon premika v Tabeli 4.1. Pretvorba formata iz stare v novo obliko (R) za **drugi sklep**, kjer je θ_{2Max} enak razponu premika v Tabeli 4.1, torej 180°:

- Če je vhodna vrednost θ_2 manjša od 0° in večja od -90°:

$$R = \left(\frac{\theta_{2Max}}{2}\right) + abs(\theta_2) \tag{4.1}$$

- Če je vhodna vrednost θ_2 večja ali enaka 0° in manjša od 90°:

$$R = \left(\frac{\theta_{2Max}}{2}\right) - \theta_2 \tag{4.2}$$

Za tretji sklep, kjer je θ_{3Max} enak 300°, velja:

- Če je vhodna vrednost θ_3 manjša od 0° ali večja od -150°:

$$R = \left(\frac{\theta_{3Max}}{2}\right) + abs(\theta_3) \tag{4.3}$$

- Če je vhodna vrednost θ_3 večja ali enaka 0° in manjša od 150°:

$$R = \left(\frac{\theta_{3Max}}{2}\right) - \theta_3 \tag{4.4}$$

Za četrti sklep, kjer je θ_{4Max} enak 120°, velja:

- Če je vhodna vrednost θ_4 manjša od 0° ali večja od -60°:

$$R = \left(\frac{\theta_{4Max}}{2}\right) + abs(\theta_4) \tag{4.5}$$

- Če je vhodna vrednost θ_4 večja ali enaka 0° in manjša od 60°:

$$R = \left(\frac{\theta_{4Max}}{2}\right) - \theta_4 \tag{4.6}$$

4.3 Izračun za drugo izbiro prvega sklepa

Če ima prvi sklep dve mogoči rešitvi (Poglavje 4.3), je potrebno izračunati tudi vrednosti sklepov za premikanje za drugo možnost (kjer se prvi sklep premakne za 180°). To izračunamo le v primeru, da zahteva podana lokacija velik premik prvega sklepa. Do tega pride običajno v primeru, da se robotska roka prestavi iz območja 2 v območje 4 ali

obratno (Slika 3.3). V tem koraku lahko pridobimo nekaj dodatnih možnosti za končno rešitev. Če ima prvi sklep dve mogoči rešitvi (Poglavje 4.3, Korak 2), se vrednosti ostalih treh sklepov izračunajo za vse na enak način. Formula za izračun druge rešitve drugega sklepa (θ'_2 na Sliki 4.2), kjer θ_{2Max} pomeni vrednost razpona drugega sklepa (Tabela 4.1):

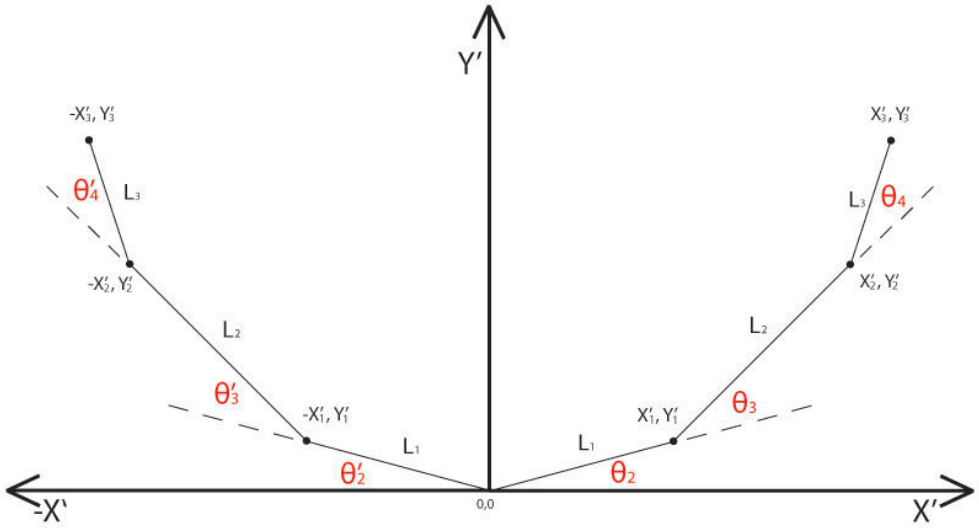
$$\theta'_2 = \frac{\theta_{2Max}}{2} + \left(\left(\frac{\theta_{2Max}}{2}\right) - \theta_2\right) \tag{4.7}$$

Formula za izračun druge rešitve tretjega sklepa, kjer θ_{3Max} pomeni vrednost razpona tretjega sklepa (Tabela 4.1):

$$\theta'_3 = \frac{\theta_{3Max}}{2} + \left(\left(\frac{\theta_{3Max}}{2}\right) - \theta_3\right) \tag{4.8}$$

Formula za izračun druge rešitve četrtega sklepa, kjer θ_{4Max} pomeni vrednost razpona četrtega sklepa (Tabela 4.1):

$$\theta'_4 = \frac{\theta_{4Max}}{2} + \left(\left(\frac{\theta_{4Max}}{2}\right) - \theta_4\right) \tag{4.9}$$



Slika 4.2: Primer pozicij sklepov prvega sklepa pred in po premiku za 180°.

4.4 Izbira končne rešitve

Algoritem iskanja končne rešitve deluje na podlagi podatkov, kje se robotska roka trenutno nahaja, in podatkov, kam se bo premaknila.

4.4.1 Izbira rešitve v primeru, da ima prvi sklep eno rešitev

Prvi sklep ima vrednost, ki jo izračunamo v Poglavju 3.4, Korak 1.

Potek izbire končne rešitve:

- Za drugi, tretji in četrti sklep pri vsaki posamezni rešitvi izračunamo, kolikšna je razlika med trenutno pozicijo (θ_2, θ_3 in θ_4), in pozicijo, kamor bi se roka lahko premaknila (θ'_2, θ'_3 in θ'_4). Te razlike poimenujemo: $\Delta\theta_2$, $\Delta\theta_3$ in $\Delta\theta_4$. Formule za drugi, tretji in četrti skelp:

$$\Delta\theta_2 = abs(\theta_2 - \theta'_2) \tag{4.10}$$

$$\Delta\theta_3 = abs(\theta_3 - \theta'_3) \tag{4.11}$$

$$\Delta\theta_4 = abs(\theta_4 - \theta'_4) \tag{4.12}$$

- Vedno preverimo, katera izmed teh treh vrednosti je največja (C_{Max}), in jo shranimo:

$$C_{Max} = \max(\Delta\theta_2, \Delta\theta_3, \Delta\theta_4) \tag{4.13}$$

- Shranjeno vrednost C_{Max} vedno primerjamo z rezultati prejšnjih vrednosti. Če je ta vrednost manjša od vseh predhodnih C_{Max} vrednosti, potem jo shranimo kot novo najmanjšo vrednost oziroma kandidata za končno rešitev.

- Tista vrednost, ki je na koncu najmanjša, je tudi končni rezultat.

Tako smo izbrali najbližjo pot premika na podano lokacijo, glede na trenutno pozicijo robotske roke. V Tabeli 4.3 je prikazana rešitev (vrstica 3), ki bi jo algoritem izbral izmed petih mogočih rešitev. Odebeljene so vse največje razdalje pri posameznem premiku, v tretji vrstici pa dobimo končno rešitev, saj je tu ta vrednost najmanjša.

Št. primera	$\Delta\theta_1$	$\Delta\theta_2$	$\Delta\theta_3$
1.	11	**34**	28
2.	**48**	30	25
3.	5	18	**28**
4.	90	45	**110**
5.	**62**	42	16

Slika 4.3: Primer podatkov za izbiro najkrajše poti in rešitve, ki je v odebeljeni vrstici.

4.4.2 Izbira končne rešitve v primeru, da ima prvi sklep dve rešitvi

Kako se izračuna **prvo mogočo rešitev** pri prvem sklepu je opisano v poglavju 3.3, **korak 1**, kako se pride do druge rešitve pa v istem poglavju, **korak 2**.

Ker imamo dve možnosti, tukaj dodatno preverjamo, kolikšna je razlika med trenutno pozicijo prvega sklepa in pozicijo, komor se mora ta sklep premakniti. Označimo z $\Delta\theta_1$ premik za prvo rešitev in $\Delta\theta_1'$ za drugo rešitev.

Postopek preverjanj:

- Preverimo, ali je $\Delta\theta$ večja od vnaprej določene konstante (mi določimo to vrednost na 135°) in se hkrati nahaja v polju 2, 3 ali 4 (Slika 3.3).

- Če pogoj iz zgornje točke drži, primerjamo vrednosti $\Delta\theta_2$, $\Delta\theta_3$ in $\Delta\theta_4$ na način, kot smo to naredili v poglavju 4.4.1. Če pogoj iz zgornje točke ne drži, drugo rešitev za prvi sklep zavržemo.

Končni rezultat je tisti, ki ima najmanjšo vrednost, kot je izpeljano v poglavju 4.4.1.

Vrednost $\Delta\theta_1$ smo primerjali s konstanto, ki smo jo postavili na 135° in ni nujno najboljša izbira. To je zgolj neka okvirno postavljena meja. Nesmiselno pa je, da bi bilo to število zelo majhno, saj bi računali veliko število neustreznih rešitev. Lahko pa bi bila ta meja tako večja, kot tudi manjša od zdajšnje.

4.5 Določanje orientacije prijemala

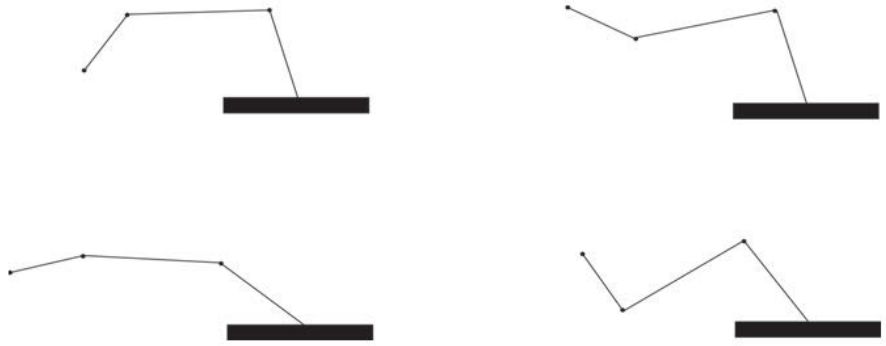
V splošnem moramo pri IK upoštevati celotno lego, torej poleg pozicije tudi orientacijo prijemala. Da robotska roka pride na neko točko obrnjena čim bolj primerno, da prime predmet, ki se nahaja na podlagi, smo prilagodili filtriranje množice rešitev. To prilagoditev se lahko vklopi opcijsko.

Prilagoditev deluje tako, da se glede na to, kam je obrnjeno prijemalo, izločijo neprimerne rešitve. Privzeta vrednost za ustrezne rešitve je, da je prijemalo vzporedno s tlemi ali pa je obrnjeno proti tlom (leva primera na sliki 4.4). Nikakor pa ne stran od tal, kot prikazujeta desna primera na isti sliki. Tako se izognemo, da roka ne pride na neko točko s tako smerjo prijemala, da je težko ali celo nemogoče prijeti predmet. Če bi bil predmet, ki ga želimo prijeti, postavljen na polico, ki je višje od izhodišča, ali bi bil predmet neobičajne oblike, bi bila leva primera (Slika 4.4) mogoče celo boljša.

Da lahko prijemalu določimo orientacijo, seštejemo pozicijo drugega, tretjega in četrtega sklepa robotske roke:

$$\theta_{Sum} = \Delta\theta_2 + \Delta\theta_3 + \Delta\theta_4 \tag{4.14}$$

To storimo, preden določimo parametre rotacije (Poglavje 4.2), saj je kasneje format sklepov drugačen in orientacije ne bi mogli tako enostavno izračunati. Tako dobimo orientacijo prijemala za množico rešitev. Pri izbiri končne rešitve (Poglavje 4.4) nato algoritem upošteva le rešitve, ki imajo ustrezno orientacijo prijemala.



Slika 4.4: Levo na sliki sta prikazana primera položajev, ki se zgodita, če je izbira za lažji prijem vklopljena, desno pa, kaj se pogosto zgodi, če je ta izbira izklopljena.

4.6 Uskladitev hitrosti premikanja sklepov

Uskladitev hitrosti premikanja je način premikanja sklepov, kjer bi se vsi sklepi ustavili hkrati. Torej, sklepi bi se premikali z različnimi hitrostmi, glede na to, kolikšna je oddaljenost do ciljne pozicije.

Uskladitev je pri robotski roki OWI-535 težko izvedljiva, predvsem zaradi dveh razlogov:

- zaradi različnih hitrosti premikanja, ne glede na to, če podamo sklepom enako hitrost. Razlika nastane zaradi karakteristik sklepov ter različnih situacij, v katerih se lahko posamezen sklep nahaja. V primeru, da se sklep premika navzgor glede na tla, pomeni že večjo obremenitev za motorček, torej tudi počasnejšo hitrost, kot, če se giblje navzdol. Mogoča **rešitev te težave** bi bila, da bi v intervalih preverjali,

kako se spreminja pozicija posameznega sklepa. V tem primeru bi lahko izračunali tudi dejansko hitrost premikanja in bi jo glede na to po potrebi spreminjali,

- ker se sklepi, predvsem tisti, ki so bolj obremenjeni (spodnji), težko premikajo oz. se sploh ne začnejo premikati, če je podana hitrost premajhna. Torej smo zelo omejeni pri hitrosti premikanja roke, saj določene sklepe lahko premikamo samo z večjimi hitrostmi.

Poglavje 5

Eksperimentalni rezultati

Eksperimentalne meritve smo izvedli, da smo dobili informacijo o tem, kako natančno je premikanje robotske roke v različnih situacijah, ter, da bi izvedeli, zakaj sploh pride do teh napak.

5.1 Eksperimentalni protokol

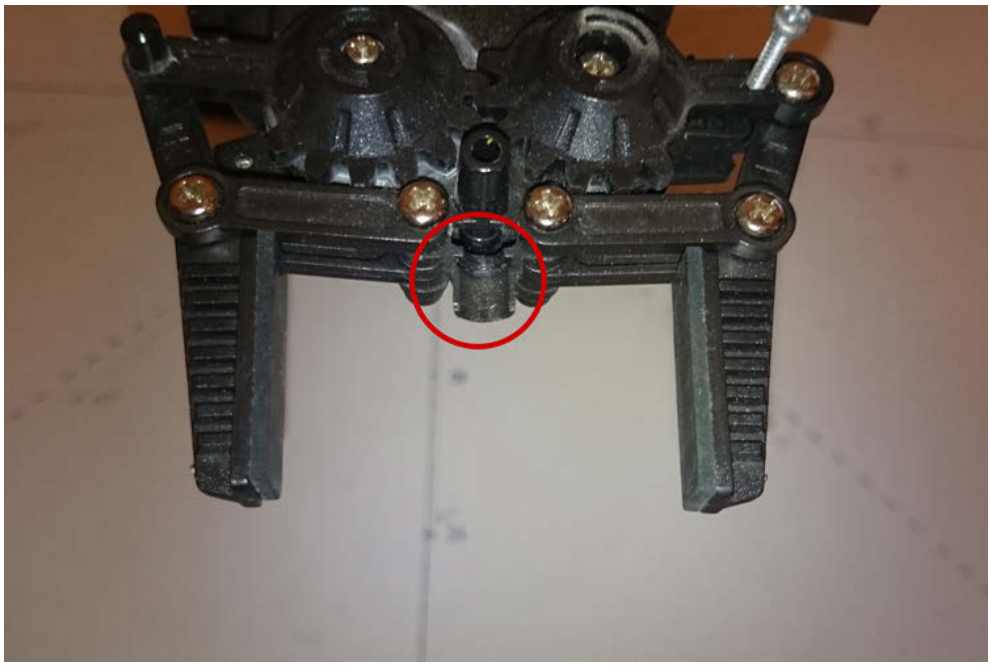
Prvi del meritev smo izvedli na način, da smo robotu podali dvajset naključnih točk, ki so v območju delovanja roke. Robotska roka se je premikala od točke do točke, pri vsaki točki pa smo izmerili dejanske vrednosti pozicije roke. V drugem delu smo določili eno fiksno točko v prostoru, nato pa smo roko iz dvajsetih naključnih točk premaknili v to točko. Meritve so se izvajale na milimeter natančno.

Da so se meritve lahko naredile, smo na večji papir narisali koordinatni sistem, v središče le tega pa postavili roko. Meritve so izvedene s pomočjo kocke, na katero je pritrjeno daljše ravnilo, ki je obrnjeno navpično. Iz tega ravnila se je lahko razbrala vrednost navpične koordinate. Ko se je ravnilo dotikalo vrha roke (Slika 5.1), smo na spodnjem delu ravnila dobili točko v prej omenjenem koordinatnem sistemu. S pomočjo drugega ravnila pa sta se iz te točke odmerili še drugi dve koordinati.

5.2 Rezultati meritev

5.2.1 Napake po posameznih koordinatah

Glede na podane koordinate in rezultate meritev izračunamo napako za koordinate X, Y, Z za vsako točko. Kot lahko vidimo na 2D (Slika 5.2) in 3D (Slika 5.3) grafu, so



Slika 5.1: Točka, na koncu robotske roke, na kateri so se izvajale meritve.

se največje napake pojavile pri Y koordinati (navpična os). Hkrati pa je bilo pri isti koordinati največ primerov brez napake. Kljub temu je povprečna napaka Y koordinate največja. Povprečne napake po posameznih koordinatah:

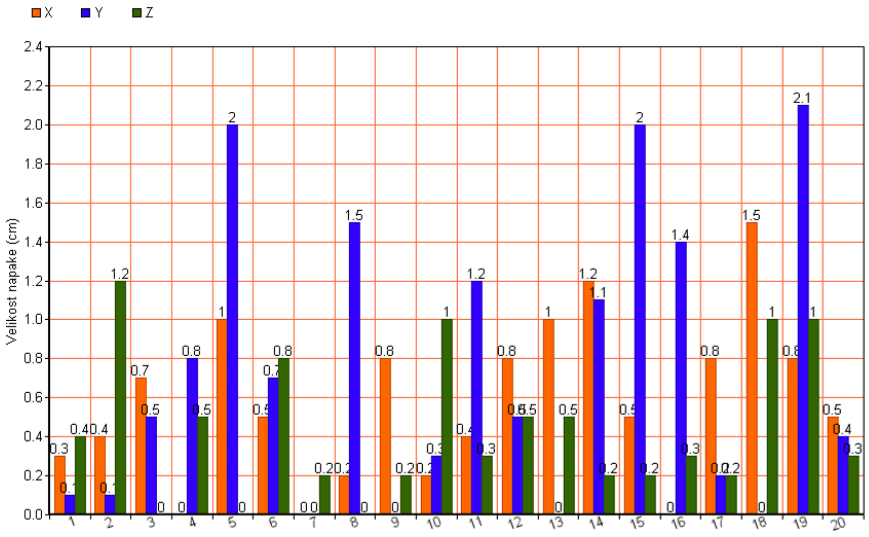
- Koordinata X: 0.58cm.
- Koordinata Y: 0.75cm.
- Koordinata Z: 0.44cm.

5.2.2 Skupna napaka

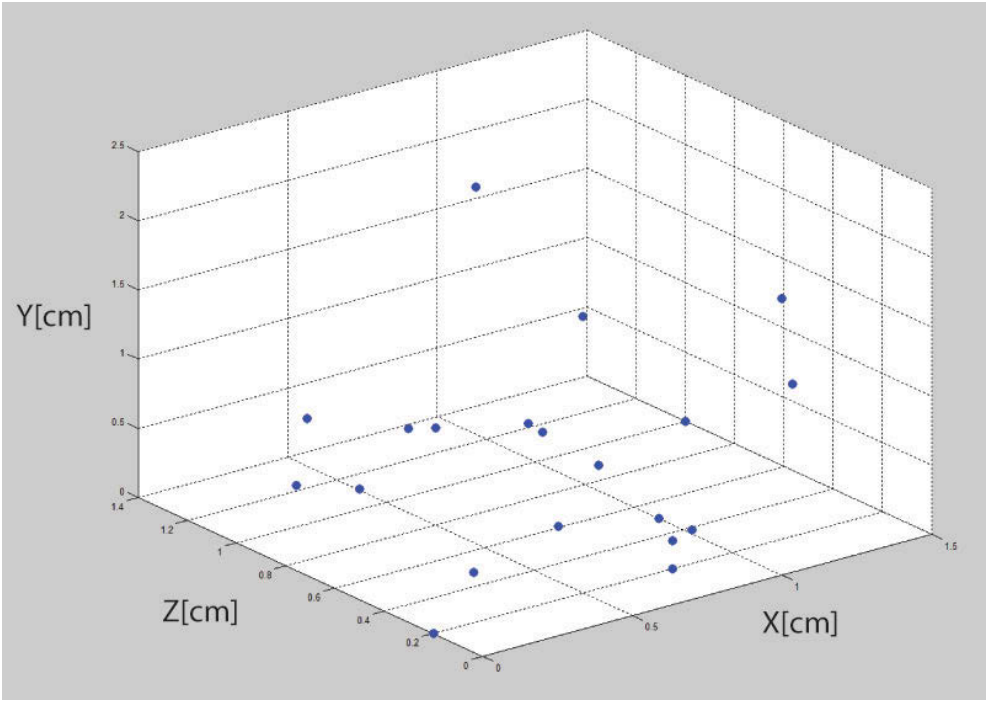
Na grafu (Slika 5.4) so prikazane evklidske razdalje med podanimi točkami ter rezultati. Meritve so se izvajale ob premikanju robotske roke iz točke v točko, v enakem zaporedju, kot so prikazane napake na Sliki 5.4. Razdalje so izračunane po formuli za izračun evklidske razdalje v 3D prostoru, kjer so X_1, Y_1 in Z_1 koordinate, ki so bile podane robotski roki, X_2, Y_2 in Z_2 pa izmerjene dejanske pozicije roke po premiku:

$$d = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2 + (Z_2 - Z_1)^2} \tag{5.1}$$

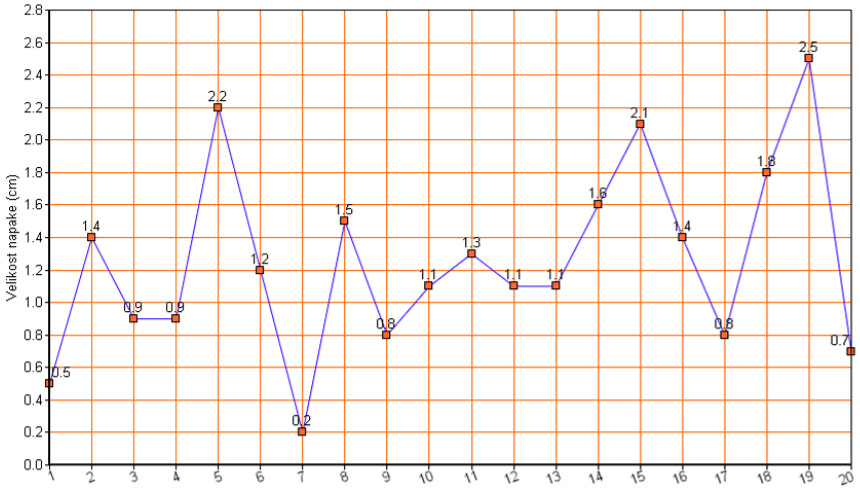
Povprečna razdalja med podano točko in točko dejanskega premika je 1.21cm.



Slika 5.2: Prikaz odstopanja posameznih koordinat X, Y in Z v 2D grafu.

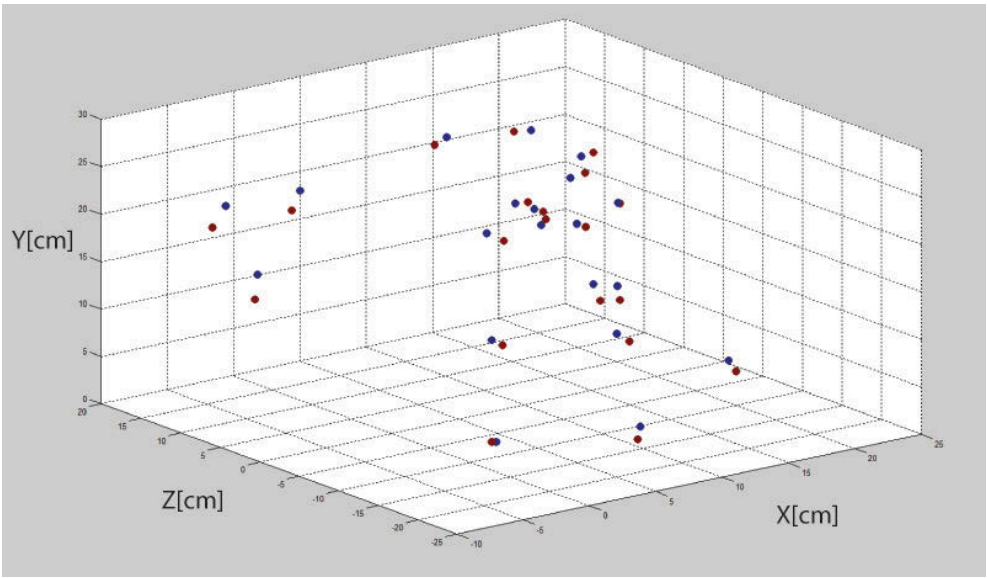


Slika 5.3: Prikaz odstopanja posameznih koordinat X, Y in Z v 3D grafu.



Slika 5.4: Prikaz napak dejanske razdalje med podano točko in izmerjenim rezultatom.

Na Sliki 5.5 je razvidna napaka glede na pozicijo v prostoru. Modre pike predstavljajo točke, ki smo jih podali robotski roki, rdeče pa izmerjene točke po premiku roke. Razdalja med najbližjo rdečo in modro piko pomeni velikost napake. Razvidno je, da je napaka na levi strani grafa nekoliko večja, kot pri drugih točkah, saj je razdalja med pari točk tu večja. Ker so odstopanja pri točkah na levi strani grafa podobna (tako razdalja kot tudi smer odstopanja), lahko predvidevamo, da nekje pride do sistematične napake.

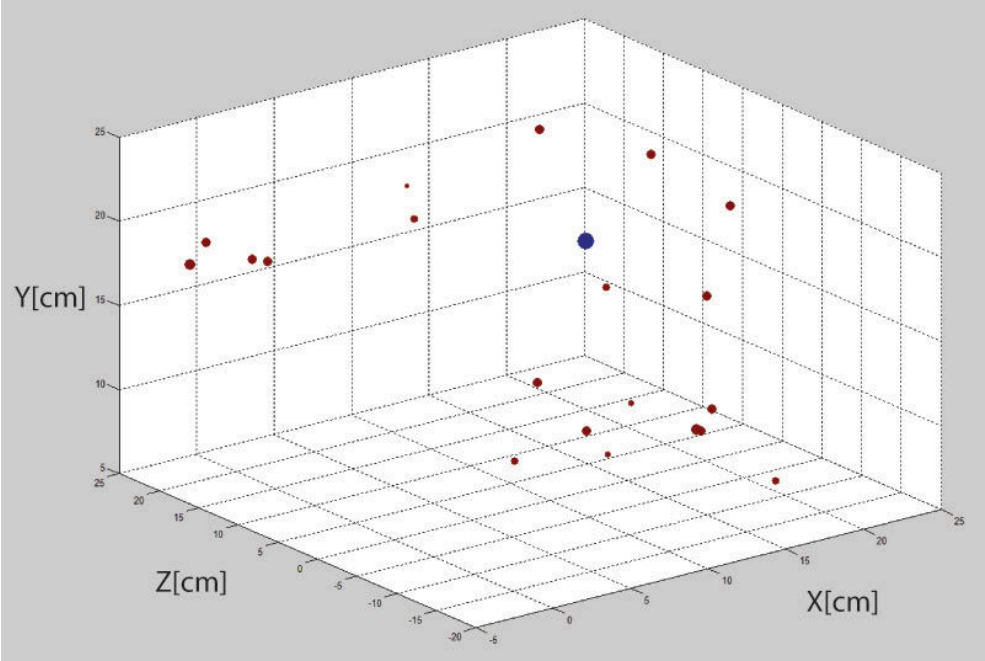


Slika 5.5: 3D prikaz napak glede na dejanske pozicije merjenih točk.

5.2.3 Napake pri premikanju v isto točko

V prostoru smo izbrali točko (modra pika na Sliki 5.6), kamor se je robotska roka premaknila večkrat zapored iz različnih pozicij. Iz pozicije rdečih pik (Slika 5.6) je razvidno, iz katere točke se je roka premaknila, velikost te pike pa je podatek za velikost napake. Večja ko je pika, večja je bila napaka pri premiku iz te točke.

Povprečna napaka pri premiku v izbrano točko je bila 0.88cm. Napaka pri premiku v isto točko je torej v tem primeru manjša od napake pri premiku v naključno točko, kjer je povprečna napaka znašala 1.21cm.



Slika 5.6: Prikaz napak pri premikanju roke iz različnih točk (rdeče pike) v vnaprej določeno točko (modra pika).

5.2.4 Diskusija

Razlog, da pride do napak, tiči najverjetneje tako pri programskem kot mehanskem delu roke. Izračun IK (Poglavje 3) je dokaj točen. Tudi preverjanje s funkcijo FK (Poglavje 3.8) pokaže, da pri samem izračunu pride le do minimalnih napak. K zmanjšanju napak pa bi verjetno pripomoglo natančnejše umerjanje (angl. calibration) vseh sklepov robotske roke. Prav zaradi odstopanja enega (mogoče tudi več) izmed sklepov najverjetneje pride do sistematične napake, ki je razvidna na Sliki 5.5.

Ker gre za nizkocenovno robotsko roko, pa k velikosti napake najverjetneje pripomore tudi sama enostavnost sestave roke, saj nam ta roka ne omogoča natančnosti kot nekatere

kakovostnejše in naprednejše robotske roke. Pri roki OWI-535 ni nič nenavadnega, če se roka ustavi z napako ene stopinje ali dveh, pri bolj kakovostnih robotskih rokah pa bi bila to že ogromna napaka.

Poglavje 6

Sklepne ugotovitve

Za robotsko roko OWI-535 je narejena osnovna verzija premikanja, ki temelji na tem, da je premikanje iz točke v točko čim bolj časovno učinkovito. Algoritem izračuna večje število možnosti, kako bi se lahko robotska roka premaknila, da preide iz ene točke v drugo. Na podlagi točke, kjer se v nekem trenutku nahaja robotska roka, ter točke, kam naj se roka premakne, se izbere najbolj primerna rešitev. Poleg določanja pozicije prijemala v prostoru, ki je glavni del diplomske naloge, je implementirana tudi možnost nastavitve orientacije prijemala. Podatek, kam naj se roka premakne, uporabnik poda prek konzole. Celotna koda diplomske naloge je napisana v programskem jeziku C++, v ogrodju ROS.

Glede na to, da je robotska roka cenovno lahko dostopna, se je roka kljub temu izkazala za dokaj natančno. Povprečna napaka pri premikanju je znašala 1.21cm. Robotska roka ima skupno dolžino segmentov 27cm. Če je roka popolnoma iztegnjena, je lahko med točkami, ki jih lahko dosega, tudi več kot pol metra razdalje. Iz tega vidika povprečna napaka 1.21cm ni zelo velika.

Z nekaj spremembami v delu, kjer poteka izbira končne rešitve iz množice rezultatov, bi se lahko prilagodilo, da je rezultat učinkovit tudi na kakšen drugačen način, ne samo časovno. S prilagajanjem zaporedja premikanja sklepov pa bi se lahko priredilo premikanje tudi na tak način, da bi se roka izognila določeni oviri. Pri rezultatih eksperimentalnih meritev lahko predvidevamo, da prihaja do sistematičnih napak pri premikanju. Zato je velika verjetnost, da bi bila robotska roka z ustreznim umerjanjem lahko še bolj natančna. Čeprav robotska roka OWI-535 ni najbolj zanesljiva, pa je zaradi cenovne dostopnosti zelo primerna za začetnike na področju robotike. Hkrati pa s štirimi sklepi za premikanje in prijemalom omogoča dokaj kompleksno gibanje. Če bi robotska roka vsebovala še del, ki je bil pri naši roki naknadno dodelan, bi bila primerna tudi za tiste, ki bi roko želeli sami programirati.

Z nadgradnjo strojne opreme, ki omogoča računalniško programiranje roke, in izdelavo programske rešitve za krmiljenje robotske roke s pomočjo inverzne kinematike je manipuliranje z robotsko roko precej olajšano in avtomatizirano. Hkrati pa je to dobra podlaga za nadaljnji razvoj sistema za manipulacijo robotske roke v 3D prostoru.

Literatura

[1] International Federation of Robotics. Dostopno na (Marec 2015):
http://www.ifr.org/

[2] International Federation of Robotics, Statistical department. Dostopno na (Marec 2015):
http://www.worldrobotics.org/

[3] KUKA Robotics. Dostopno na (Marec 2015):
http://www.kuka-robotics.com/

[4] Google globoko zakorakal v robotiko. Dostopno na (Marec 2015):
http://www.dnevnik.si/magazin/znanost-in-tehnologija/google-globoko-zakorakal-v-robotiko

[5] Arduino Mega. Dostopno na (Marec 2015):
http://arduino.cc/en/Main/arduinoBoardMega

[6] OWI-535 manual. Dostopno na (Marec 2015):
http://www.robotshop.com/media/files/pdf/owi-535_manual.pdf

[7] Bjarne Stroustrup, The C++ Programming Language, 1986.

[8] Bjarne Stroustrup, A Tour of C++, 2013.

[9] ROS Documentation. Dostopno na (Marec 2015):
http://wiki.ros.org/

[10] ROS. Dostopno na (Marec 2015):
http://www.ros.org/

[11] Tadej Bajd, Osnove robotike, Založba FE in FRI, 2006.

[12] Anže Rezelj, Predelava robotske roke OWI 535 za študijske namene, Tehnično poročilo TR-VICOS-2014-11, 2014.